Projet TAL

DETECTION AUTOMATIQUE D'OPINIONS DANS DES TWEETS FRANÇAIS

Sharleyne LEFÈVRE
MASTER 2 LOUTAL | UNIVERSITE D'ORLEANS | 2018-2019

Table des matières

1 Présentation du projet

Le projet que nous avons pu mener sur la détection et l'annotation automatique des émotions et des opinions dans des tweets français, est issu de la tâche 3 du DeFT 2018. Le DeFT est un atelier annuel d'évaluation francophone en fouille de textes. Cette tâche consiste en l'annotation des marqueurs de sentiment dans les tweets, la cible de l'expression du sentiment et les modifieurs (négation, amplification, etc.). Nous avons décidé, pour ce projet, de nous focaliser sur l'annotation automatique des marqueurs de sentiment. Pour réaliser notre tâche, nous avons décidé d'utiliser une méthode à base d'apprentissage automatique.

2 Etat de l'art

2.1 DeFT 2015

Des travaux ont déjà été menés auparavant, notamment au Défi Fouille de Textes (DeFT) 2015. A l'origine ce projet a été appliqué sur des tweets portant sur le changement climatique.

La première tâche consistait à classifier les tweets selon leur polarité avec les classes : positif, négatif, neutre ou mixte (si le tweets contenait un sentiment positif accompagné d'un sentiment négatif).

La deuxième tâche consistait en une classification plus fine des tweets, elle se découpe en deux parties. La première partie étant de classifier les tweets en 4 classes qui sont : la classe « INFORMATION » pour les tweets qui contiennent un fait réel et observable, la classe « OPINION » pour les tweets qui expriment une opinion, la classe « SENTIMENT » pour les tweets qui contiennent un sentiment, et enfin, la classe « EMOTION » pour les tweets qui comportent une émotion. Il faut alors distinguer l'opinion, le sentiment et l'émotion. L'opinion désigne une manière de penser sur un sujet ou un ensemble de sujets, jugement personnel que l'on porte sur une question, qui n'implique pas que ce jugement soit obligatoirement juste[1]. Le sentiment désigne la faculté de sentir, de percevoir une sensation[2] contrairement à l'émotion qui désigne une conduite réactive, réflexe, involontaire vécue simultanément au niveau du corps d'une manière plus ou moins violente et affectivement sur le mode du plaisir ou de la douleur[3]. La deuxième partie de la tâche 2 consistait à préciser la nature spécifique des trois classes liées à la perception des choses (opinion, sentiment, émotion). 18 classes proposées antérieurement dans le projet uComp[4], ont été utilisées pour identifier la spécificité des classes précédemment citées : COLÈRE, PEUR, TRISTESSE, DÉGOÛT, ENNUI, DÉRANGEMENT, DÉPLAISIR, SURPRISE NÉGATIVE, APAISEMENT, AMOUR, PLAISIR, SURPRISE POSITIVE, INSATISFACTION, SATISFACTION, ACCORD, VALORISATION, DÉSACCORD, et DÉVALORISATION. Il peut arriver qu'un tweet possède plusieurs émotions, ainsi, ce sera la classe dominante qui l'emportera.

La troisième tâche consistait à identifier les groupes et les relations dans les tweets. Les groupes étant : la « SOURCE » qui est la personne qui exprime l'opinion/sentiment/émotion, la « CIBLE » qui est l'objet de l'opinion/sentiment/émotion, l' « OSEE » qui est l'expression de l'opinion/sentiment/émotion, annoté grâce aux 18 classes précédemment présentées, le « MODIFIEUR » qui correspond aux adverbes de modification comme « très », « trop », etc, et la

[1] http://cnrtl.fr/definition/opinion
[2] http://cnrtl.fr/definition/sentiment
[3] http://cnrtl.fr/definition/%C3%A9motion
[4] https://www.ucomp.eu/

« NEGATION » pour les marqueurs de négation comme « ne », « pas », « ni », etc. L'identification des relations dans les tweets permettent de mettre en rapport les éléments cités précédemment entre eux : « DIT » met en relation la SOURCE avec l'OSEE, « SUR » met en relation l'OSEE avec la CIBLE, « MOD » met en relation le modifieur avec l'OSEE et « NEG » met en relation le marqueur de négation avec l'OSEE.

Les participants ont utilisé des méthodes à base d'apprentissage supervisé. L'apprentissage supervisé (ou supervised learning) consiste à utiliser des données annotées pour entraîner un modèle, les données sont déjà associées à des labels ou des classes cibles. Les participants ont utilisé les algorithmes suivants : SVM (LIRMM, LINA/Dictanova, LINA–Dimeco), Naïve Bayes (IRISA, LIMSI), réseau de neurones (IRISA, Proxem), algorithme de prédiction par correspondance partielle : PPMC (TU Moldova), et autres modèles probabilistes (LIF). La dernière équipe a choisi d'utiliser une chaîne de traitements TextAnalyst composée de lexiques et de plusieurs modules qui s'enchainent en cascade.

En ce qui concerne la première partie de la tâche 2, les résultats les plus satisfaisants en terme de macro-précision ont été produits par le LIRMM avec la méthode d'apprentissage supervisé avec le classifieur SVM. Ces derniers ont obtenus une macro-précision de 0.613.

Équipe	Soumissions			Classement
LIRMM (équipe 17)	**0,613**	0,563	0,552	1
INaLCO (équipe 2)	0,572	0,562	**0,575**	2
IRISA (équipe 14)	**0,572**	0,478	0,502	3
LIF (équipe 3)	0,558	**0,560**	0,535	4
LINA / Dictanova (équipe 6)	0,508	**0,514**		5
TU Moldova (équipe 22)	**0,383**	0,382		6
IRIT / LIMSI (équipe 10)	0,269	**0,332**	**0,332**	7
LINA–Dimeco (équipe 25)	0,000	0,000	**0,097**	8
(équipe 4)	0,029	**0,029**		9

Figure 1 - Résultats de la première partie de la tâche 2 du DeFT 2015

Sur la deuxième partie de la tâche 2, c'est le LIF qui l'emporte avec une macro-précision de 0.347 en ayant utilisé plusieurs modèles probabilistes.

Équipe	Soumissions			Classement
LIF (équipe 3)	**0,347**	0,327	0,327	1
INaLCO (équipe 2)	**0,337**	0,292	0,304	2
IRISA (équipe 14)	**0,325**	0,258	0,316	3
TU Moldova (équipe 22)	**0,226**	0,175		4
LIRMM (équipe 17)	0,037	**0,174**	0,007	5
LINA / Dictanova (équipe 6)	**0,028**	0,027		6
(équipe 4)	0,002	**0,002**		7
LINA–Dimeco (équipe 25)	0,000	0,000	0,000	8

Figure 2 - Résultats de la deuxième partie de la tâche 2 du DeFT 2015

2.2 DeFT 2018

Nous pouvons également expliquer en quoi ont consisté les tâches du DeFT 2018, dont une des tâches proposées a été repris lors de ce projet de TAL. Nous pouvons rappeler que pour la campagne 2018, la thématique des tweets était « les transports ».

La première tâche consistait à une classification des tweets en « TRANSPORT » et « NON-TRANSPORT » afin de faire un tri sur les tweets qui seront traités. La deuxième tâche reprend la première tâche du DeFT 2015 qui est de déterminer la polarité des tweets en « NEGATIF »,

« POSITIF », « NEUTRE » et « MIXPOSNEG ». La troisième tâche est celle qui nous intéresse, elle reprend la tâche 2 du DeFT 2015. Il s'agit de déterminer, pour chaque expression, les marqueurs d'opinion/sentiment/émotion et leur cible. La dernière tâche étant encore exploratoire, nous n'allons pas la détailler. Elle consiste à obtenir une annotation complète : déterminer pour chaque expression de sentiment l'empan de texte minimal référrant à l'expression de sentiment et les empans de texte maximaux référrant respectivement à la CIBLE du sentiment, c'est à dire à l'objet qu'il concerne, et à la SOURCE, c'est à dire à l'entité qui exprime ce sentiment.[5]

Les méthodes utilisées pour ces deux campagnes ont été principalement basées sur de l'apprentissage automatique. La majorité des systèmes utilisés par les participants repose sur des approches d'apprentissage supervisé. Ces méthodes semblent être de bons choix aux vues de notre corpus déjà annoté.

Pour notre projet qui reprend la tâche 3 du DeFT 2018, nous avons décidé d'utiliser également une méthode à base d'apprentissage automatique supervisé, en complément de CRF. Les CRF n'ayant pas été utilisés lors du DeFT 2015 semble être une bonne piste pour la tâche 3 du DeFT 2018.

Les CRF (Conditional Random Fields ou Champs Markoviens Conditionnels) sont des algorithmes de classification se basant sur des modèles probabilistes discriminants. Il s'agit qu'un élément arrive après un autre selon des probabilités. Les CRF permettent d'apprendre à annoter des données en se basant sur un ensemble d'exemples déjà annotés. Ils ont donné d'excellents résultats sur différentes tâches pouvant se ramener à l'étiquetage d'unités linguistiques[6] (la reconnaissance d'entités nommées ou le part-of-speech tagging par exemple). C'est une méthode qui semble intéressante car les CRF prennent en compte les dépendances entre les étiquettes (quelle est l'étiquette précédente pour lui donner un indice). Les CRF cherchent la meilleure séquence d'étiquettes pour classer les tokens.

3 Corpus

Nous avons travaillé sur un corpus de 20 292 tweets français au total, sur la thématique des transports en communs. Ce corpus était déjà annoté avec les mêmes conventions que pour le DeFT 2015. Comme nous travaillons sur des tweets, nous avons conscience que nous pouvons avoir des fautes de frappes ou d'orthographe, des smileys, des expressions familières, etc. L'annotation par un humain permet de contourner ces difficultés car l'humain comprend de quoi il s'agit. Cependant, notre hypothèse est qu'avec l'apprentissage automatique, il se pourra que certaines fautes d'orthographe n'aient pas été rencontrées auparavant et ne permette pas la détection de l'opinion/sentiment/émotion.

La préparation de notre corpus de 20 292 tweets a nécessité un découpage. Il s'agit ici de découper le corpus en 2 parties afin d'obtenir un corpus d'entrainement, un corpus assez volumineux qui sera utilisé pour entrainer l'algorithme, un corpus de test, moins volumineux que le premier, qui permettra de tester l'algorithme sur un corpus encore jamais vu par la machine. Nous avons donc procédé à un découpage avec les proportions suivantes :

- Corpus d'entrainement : 70%,
- Corpus de test : 30%,

[5] https://deft.limsi.fr/2018/#
[6] http://www.univ-orleans.fr/lifo/Members/duchier/equipe/pmwiki/pmwiki.php?n=TAL.CRF

En principe le découpage inclus un troisième découpage qui est le corpus d'évaluation qui vaut 10% du corpus total. Le tutoriel que nous avons suivi n'en comportait pas. C'est pour cette raison que nous n'avons pas de corpus d'évaluation.

Le corpus a été annoté avec le format IOB. Pour chaque entité annotée, nous avons une indication sur la place qu'a l'élément dans l'annotation : B pour Beginning indique le début d'une séquence I pour Inside indique un mot à l'intérieur d'une séquence et O pour Outside indique un mot qui n'est pas dans la séquence. Nous pouvons donner un exemple :

Pourquoi	0	7	O
c'	8	10	O
est	11	13	O
toujours	15	22	O
n'	**23**	**25**	**B-DEVALORISATION**
importe	**26**	**32**	**I-DEVALORISATION**
quoi	**34**	**37**	**I-DEVALORISATION**
le	39	40	O
départ	42	47	O
des	49	51	O
bus	53	55	O
26	57	58	O
à	60	60	O
Nation	62	67	O
@GroupeRATP	69	79	O
?	81	81	O
#mystère	83	90	O

Sur cet exemple nous voyons bien qu'il n'y a qu'un mot par ligne et le format IOB nous permet de récupérer la séquence entière qui marque la dévalorisation.

Comme nous avons décidé de ne traiter que les marqueurs d'opinion/sentiment/émotion, nous avons retiré les autres classes qui ne font pas partie des 18 classes citées plus haut, comme la classe CIBLE par exemple. Pour cela nous avons utilisé une expression régulière permettant de remplacer les classes « indésirables » « O ».

4 Développement d'un programme

4.1 Modules et découpage du corpus

Pour réaliser ce projet, nous avons utilisé le langage de programmation Python ainsi que des modules adaptés à notre tâche. Nous avons dit précédemment que nous avons utilisé une méthode d'apprentissage automatique utilisant des CRF. Il s'agit de développer et d'entraîner un système à partir des données annotées qui leur nous sont fournies afin d'avoir une annotation automatique en finalité. Le système s'entraîne sur notre sous-corpus d'entrainement qui représente 70% du corpus total. Ensuite, nous devons tester le système sur un sous-corpus qui n'a jamais été vu du système, nous avons utilisé notre sous-corpus de test qui représente 30% du corpus total.

Le module que nous avons utilisé pour développer notre programme est sklearn_crfsuite (issu du module scikit_learn, un module d'apprentissage automatique en langage Python). Ne sachant pas comment procéder pour développer le programme, nous avons suivi un tutoriel[7].

4.2 Corpus du tutoriel et étapes

Le tutoriel que nous avons utilisé est issu d'un travail dont la tâche commune était la reconnaissance d'entités nommées indépendantes dans la langue, dans les données du CoNLL-2002 qui est un jeu de données au format CoNLL (similaire à nos données en IOB). CoNLL-2002 est composé de 6 fichiers en espagnol (articles d'une agence de presse) et en allemand (4 éditions du journal belge « De Morgen »).

Les entités nommées portent sur des noms de personnes (noté PER), d'organisations (ORG), de localisation (LOC), et notamment sur des informations diverses (MISC) telles que le temps et la quantité. Comme pour le DeFT, plusieurs participants au CoNLL-2002 ont travaillé sur cette tâche de reconnaissance d'entités nommées en utilisant les données pour créer un programme incluant de l'apprentissage automatique.

Le tutoriel reprend 4 étapes nécessaires à l'exécution du programme, à savoir : la collecte des données du corpus, la création/extraction des traits, l'entrainement et l'évaluation des résultats. Le programme est donné tel quel, il suffit d'adapter le script à notre corpus. Le tutoriel propose d'autres étapes après l'évaluation comme l'optimisation, il s'agit d'améliorer la qualité en essayant de sélectionner les paramètres de régularisation à l'aide d'une recherche aléatoire et d'une validation croisée en trois étapes. Cette étape étant facultative et très coûteuse en temps, nous avons décidé de nous arrêter à l'étape d'évaluation.

4.3 Formatage des données

Comme les données sur le tutoriel étaient similaires à nos données (format IOB), il a fallu avoir la même mise en forme des données pour que le tutoriel fonctionne avec nos tweets. Pour illustrer nos propos, nous pouvons donner un exemple. Dans le tutoriel, tous les tweets ont ce format-là :

Chaque tweet est contenu dans une liste, avec à l'intérieur de cette liste, un tuple par mot. Chaque tuple est constitué du mot et des paramètres associés, ici, la catégorie grammaticale et l'annotation en IOB des opinions/sentiments/émotions (le label). Pour nos données nous avons suivi le même formatage de sortie.

Données du Tutoriel	**Nos données**
[('Melbourne', 'NP', 'B-LOC'), ('(', 'Fpa', 'O'), ('Australia', 'NP', 'B-LOC'), (')', 'Fpt', 'O'), (',', 'Fc', 'O'), ('25', 'Z', 'O'), ('may', 'NC', 'O'), ('(', 'Fpa', 'O'), ('EFE', 'NC', 'B-ORG'), (')', 'Fpt', 'O'), ('.', 'Fp', 'O')]	[('par', 'PRP', 'O'), ('contre', 'PRP', 'O'), ('le', 'DET:ART', 'O'), ('trafic', 'NOM', 'O'), ('ça', 'PRO:DEM', 'O'), ('rend', 'VER:pres', 'O'), ('ouf', 'INT', 'B-NEGATIF'), (',', 'PUN', 'O'), ('on', 'PRO:PER', 'O'), ('va', 'VER:pres', 'O'), ('prendre', 'VER:infi', 'O'), ('le', 'DET:ART', 'O'),

[7] https://sklearn-crfsuite.readthedocs.io/en/latest/tutorial.html

	('métro', 'ADJ', 'O'), ('ça', 'PRO:DEM', 'O'), ('va', 'VER:pres', 'O'), ('aller', 'VER:infi', 'O'), ('100x', 'NOM', 'O'), ('plus', 'ADV', 'O'), ('vite.', 'ADJ', 'O')]

Comme pour l'étape de récupération des tweets, il était également nécessaire d'avoir un formatage similaire à celui du tutoriel au niveau des traits. Nous verrons plus de détails sur les traits dans la partie 4.5). Chaque mot renvoie un dictionnaire de traits, la valeur étant le trait, la clé étant un booléen ou des informations telles que la catégorie grammaticale, le mot ou le lemme). Nous avons adapté ce format de sortie pour nos données.

Données du tutoriel	**Nos données**
{'+1:postag': 'Fpa', '+1:postag[:2]': 'Fp', '+1:word.istitle()': False, '+1:word.isupper()': False, '+1:word.lower()': '(', 'BOS': True, 'bias': 1.0, 'postag': 'NP', 'postag[:2]': 'NP', 'word.isdigit()': False, 'word.istitle()': True, 'word.isupper()': False, 'word.lower()': 'melbourne', 'word[-2:]': 'ne', 'word[-3:]': 'rne'}	{'word.lower()': 'bon', 'word.isupper()': False, 'word.istitle()': False, 'postag': 'ADJ', 'lemme': 'bon', 'polarity': 'positive', 'joy': '0', 'fear': '0', 'sadness': '0', 'anger': '0', 'surprise': '1', 'disgust': '0'}

4.4 Utilisation de TreeTagger

Après le formatage des tweets, en vue de créer les traits pour l'apprentissage, nous avons procédé à une lemmatisation de chaque terme. Pour cela, nous avons utilisé un étiqueteur syntaxique qui se nomme TreeTagger. Nous avons utilisé TreeTagger dans Python en utilisant la commande TreeTagger faisant appel au programme dans son fichier de destination en donnant un fichier à étiqueter et un fichier de sortie (dans lequel il y aura le résultat de l'étiquetage) (Figure 3).

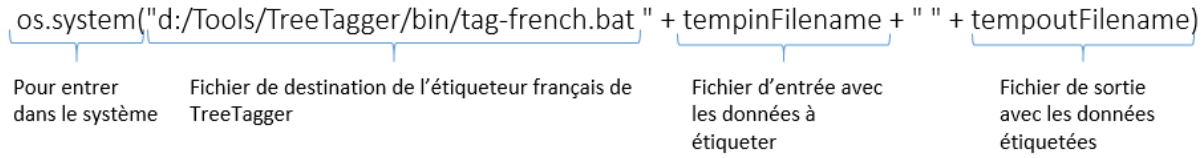

Figure 3 - Commande TreeTagger dans Python

Nous obtenons donc dans notre fichier de sortie le lexème à gauche, la catégorie grammaticale au milieu et le lemme à droite (forme canonique du mot) :

```
on      PRO:PER         on
s'      PRO:PER         se
est     VER:pres        être
trompé          VER:pper        tromper
de      PRP     de
bus     NOM     bus
```

L'étiquetage syntaxique nous permet donc de récupérer des informations qui nous servirons dans la création des traits : ajout de la catégorie grammaticale et du lemme à nos traits.

4.5 Création des traits

A la suite des étapes précédentes, nous pouvions donc passer à la création des traits (ou features). Les traits sont nécessaires pour l'entrainement des données. Ils permettent de nous donner des informations sur les lexèmes comme sa catégorie grammaticale, son écriture en minuscule, ainsi que des booléens permettant de savoir si le mot d'origine était un nombre ou si il commençait par une majuscule. Pour nos données, nous avons décidé d'utiliser les traits suivants : le mot en minuscule, deux booléens pour savoir si le mot était en majuscule à l'origine et s'il commençait par une majuscule (nom propre), la catégorie grammaticale et le lemme. S'ajoutant à ces traits, les traits caractérisant les termes porteurs de sentiments :

Pour savoir si un mot était un sentiment/une opinion/une émotion, nous avons décidé d'utiliser une liste de termes porteurs de sentiments[8] qui se présente au format CSV. Ensuite, nous avons fait une comparaison de termes (termes des tweets avec termes de la liste CSV) pour ensuite extraire la polarité (positive, négative) ainsi que les booléens associés aux sentiments joie, peur, tristesse, colère, surprise, dégoût.

id	word	polarity	joy	fear	sadness	anger	surprise	disgust
8	abominable	negative	0	1	0	0	0	1
9	abrupt	negative	0	1	0	0	0	0
10	absent	negative	0	1	1	0	0	0

Figure 4 - Extrait de la liste FEEL

N'ayant pas utilisé les fonctions d'extraction des features proposées par le tutoriel, nous nous sommes rendus compte, à l'étape d'entrainement, que toute la suite du script était construite par rapport à ces fonctions. Nous avons dû définir les variables X_train, y_train, X_test et y_test avec nos données. Les X attendant le même format que notre variable output (voir programme) et y attendant les labels contenus dans notre variable res (voir programme).

Les variables X_train et X_test doivent contenir, selon le tutoriel, une liste qui comporte pour chaque tweet une liste comportant elle-même un dictionnaire de traits par mot :

[[{'word.lower()': 'jcrois', 'word.isupper()': False, 'word.istitle()': True, 'postag': 'NAM', 'lemme': 'Jcrois'}, {'word.lower()': 'le', 'word.isupper()': False, 'word.istitle()': False, 'postag': 'DET:ART', 'lemme': 'le'}, {'word.lower()': 'chauffeur', 'word.isupper()': False, 'word.istitle()': False, 'postag': 'NOM', 'lemme': 'chauffeur', 'polarity': 'positive', 'joy': '0', 'fear': '0', 'sadness': '0', 'anger': '0', 'surprise': '0', 'disgust': '0'}, …], [{…}], …]

Les variables Y_train et y_test doivent contenir une liste qui comporte pour chaque tweet une liste comportant elle-même pour chaque mot son label. Pour cela nous avons sélectionné pour chaque tuple de chaque tweet[9], l'élément en position 2, c'est-à-dire le label, en reconstruisant le format grâce à des listes :

[['O', 'B-NEGATIF', 'O', …], ['O', 'O', 'B-PLAISIR', 'I-PLAISIR', 'O', …], …]

[8] http://www.lirmm.fr/~abdaoui/FEEL

[9] Ayant cette forme : (« Mot », « PosTag », « Label »)

Tutoriel

```
def word2features(sent, i):
    word = sent[i][0]
    postag = sent[i][1]

    features = {
        'bias': 1.0,
        'word.lower()': word.lower(),
        'word[-3:]': word[-3:],
        'word[-2:]': word[-2:],
        'word.isupper()': word.isupper(),
        'word.istitle()': word.istitle(),
        'word.isdigit()': word.isdigit(),
        'postag': postag,
        'postag[:2]': postag[:2],
    }
    if i > 0:
        word1 = sent[i-1][0]
        postag1 = sent[i-1][1]
        features.update({
            '-1:word.lower()': word1.lower(),
            '-1:word.istitle()': word1.istitle(),
            '-1:word.isupper()': word1.isupper(),
            '-1:postag': postag1,
            '-1:postag[:2]': postag1[:2],
        })
    else:
        features['BOS'] = True

    if i < len(sent)-1:
        word1 = sent[i+1][0]
        postag1 = sent[i+1][1]
        features.update({
            '+1:word.lower()': word1.lower(),
            '+1:word.istitle()': word1.istitle(),
            '+1:word.isupper()': word1.isupper(),
            '+1:postag': postag1,
            '+1:postag[:2]': postag1[:2],
        })
    else:
        features['EOS'] = True

    return features


def sent2features(sent):
    return [word2features(sent, i) for i in range(len(sent))]

def sent2labels(sent):
    return [label for token, postag, label in sent]

def sent2tokens(sent):
    return [token for token, postag, label in sent]

X_train = [sent2features(s) for s in train_sents]
y_train = [sent2labels(s) for s in train_sents]

X_test = [sent2features(s) for s in test_sents]
y_test = [sent2labels(s) for s in test_sents]
```

Notre programme

```
def preApprentissage(fichier):
    sents = traitement(fichier)
    X = sents[1] # sents[1] = output

  # pour n'avoir que les labels de chaque mot de chaque tweet
    labelsOnly = []
    for tweet in sents[0]: # sents[0] = res
        labelsTweet = []
        for mot in tweet:
            labelsTweet.append(mot[2])
        labelsOnly.append(labelsTweet)

    Y = labelsOnly # Y pour uniquement les labels (IOB)

    return (X, Y)

# traits / labels pour chaque fichier
trainData = preApprentissage(train)
testData = preApprentissage(test)

# comme dans le tuto
X_train = trainData[0] #outuput (traits)
y_train = trainData[1] #res (labels)
X_test = testData[0]
y_test = testData[1]
```

4.6 Entrainement des données

Pour entrainer les données, nous n'avons rien changé à l'algorithme proposé par le tutoriel. Nous utilisons l'algorithme d'apprentissage L-BFGS (par défaut) avec la régularisation Elastic Net (méthode statistique) (L1 + L2). L'algorithme Limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) est un algorithme d'optimisation pour paramétrer l'estimation en apprentissage automatique.

5 Évaluation

Pour l'évaluation, comme pour l'entrainement, nous n'avons rien changé à l'algorithme proposé dans le tutoriel.

Les résultats exprimés nous donnent, pour chaque label, sa précision, son rappel et son F-mesure et la dernière colonne est dédiée au support. La précision mesure le nombre d'éléments correctement annotés par rapport au nombre total d'éléments annotés. Le rappel mesure le nombre d'éléments correctement annotés par rapport au nombre d'éléments annotés dans la référence. Le F-mesure est l'indicateur de synthèse pour évaluer les algorithmes de classification de données textuelles à partir du rappel et de la précision. Plus le score s'approche de 1, plus le résultat est satisfaisant. Nous n'avons pas d'informations sur ce que représente la colonne support, mais nous pouvons supposer que cela fait référence au nombre d'entités annotées par le système.

Dans le tutoriel, les labels « O » ne sont pas pris en compte par le F-mesure. Il est dit dans le tutoriel : « Le jeu de données contient beaucoup plus d'entités annotées « O », mais nous sommes plus intéressés par les autres entités. Pour tenir compte de cela, nous utiliserons le F-mesure moyen pour calculer toutes les étiquettes sauf le « O ». ».

```
labels = list(crf.classes_)
labels.remove('O')
print(labels)
['B-LOC', 'B-ORG', 'B-PER', 'I-PER', 'B-MISC', 'I-ORG', 'I-LOC', 'I-MISC']
```

Nous obtenons donc une précision moyenne de 60%, un rappel moyen de 45% et un F-mesure moyen de 51%. Nous pouvons observer, sur la figure 5, que certains labels obtiennent de bons résultats. A contrario, nous pouvons obtenir des labels obtenant pour la précision, le rappel et le F-mesure, un score de 0,000.

	precision	recall	f1-score	support
B-ACCORD	0.414	0.167	0.238	72
I-ACCORD	0.294	0.179	0.222	28
B-AMOUR	0.702	0.325	0.444	123
I-AMOUR	0.000	0.000	0.000	12
B-APPAISEMENT	0.600	0.225	0.327	80
I-APPAISEMENT	0.400	0.143	0.211	28
B-COLERE	0.608	0.380	0.468	502
I-COLERE	0.554	0.452	0.498	398
B-DEPLAISIR	0.456	0.172	0.250	151
I-DEPLAISIR	0.345	0.211	0.262	90
B-DERANGEMENT	0.656	0.531	0.587	499
I-DERANGEMENT	0.161	0.071	0.099	126
B-DESACCORD	0.854	0.596	0.702	275
I-DESACCORD	0.368	0.125	0.187	56
B-DEVALORISATION	0.579	0.518	0.547	628
I-DEVALORISATION	0.425	0.274	0.333	237
B-ENNUI	0.667	0.214	0.324	28
I-ENNUI	1.000	0.276	0.432	29
B-INSATISFACTION	0.626	0.452	0.525	356
I-INSATISFACTION	0.227	0.112	0.150	134
B-MEPRIS	0.690	0.505	0.583	515
I-MEPRIS	0.520	0.356	0.423	216
B-MIXPOSNEG	0.000	0.000	0.000	1
B-NEGATIF	0.652	0.535	0.588	2126
I-NEGATIF	0.108	0.059	0.077	522
B-PEUR	0.784	0.609	0.686	215
I-PEUR	0.200	0.105	0.138	38
B-PLAISIR	0.882	0.741	0.805	1195
I-PLAISIR	0.416	0.244	0.308	131
B-POSITIF	0.567	0.279	0.374	501
I-POSITIF	0.402	0.174	0.243	236
B-SATISFACTION	0.830	0.646	0.727	432
I-SATISFACTION	0.269	0.073	0.115	96
B-SURPRISE_NEGATIVE	0.400	0.143	0.211	42
I-SURPRISE_NEGATIVE	0.000	0.000	0.000	13
B-SURPRISE_POSITIVE	0.833	0.185	0.303	27
I-SURPRISE_POSITIVE	1.000	0.200	0.333	10
B-TRISTESSE	0.713	0.478	0.572	224
I-TRISTESSE	0.231	0.110	0.149	82
B-VALORISATION	0.671	0.563	0.612	1268
I-VALORISATION	0.102	0.026	0.042	230
avg / total	0.605	0.450	0.511	11972

Figure 5 - Résultats de l'évaluation

La figure 5 nous montre les résultats de l'évaluation, nous avons coloré les scores les plus élevés. En bleu nous avons les scores qui s'élèvent à 100%, en orange les scores correspondant aux scores compris entre 70% et 99%, en violet les scores compris entre 50% et 69% et en jaune, nous avons les scores s'élevant à 0%. Les labels correctement annotés par rapport au nombre total d'éléments annotés obtenant une bonne précision sont : I-ENNUI, I-SURPRISE_POSITIVE, B-AMOUR, B-DESACCORD, B-PEUR, B-PLAISIR, B-SATISFACTION, B-SURPRISE-POSITIVE, B-TRISTESSE (Coloration bleue et orange). Les labels correctement annotés par rapport au nombre d'éléments annotés dans la référence ayant un rappel élevé est : B-PLAISIR (orange). Enfin, les labels ayant un F-mesure élevé sont : B-DESACCORD, B-PLAISIR et B-SATISFACTION (orange). En revanche, il y a eu un échec au niveau de la détection des labels : I-AMOUR, B-MIXPOSNEG et I-SURPRISE_NEGATIVE (jaune), ce qui fait chuter la moyenne finale. Les résultats en coloration violette restent satisfaisants par rapport à l'ensemble.

Certains labels ont pour résultat un score de 0,000 et d'autres 1,000 :

B-MIXPOSNEG (0,000) → Dans le fichier d'entrainement il n'y a que deux occurrences de B-MIXPOSNEG qui sont « monstrueux » et « malaise » et dans le fichier de test il n'y a qu'une occurrence. Il y a trop peu de données pour que le système apprenne correctement. De plus, la classe MIXPOSNEG est assez complexe et ambigüe car tout dépend du contexte des mots dans le tweet.

I-SURPRISE_NEGATIVE (0,000) → Les termes annotés dans le fichier d'entrainement sont : « choquant », « choque », « perplexe », « putain », « surréaliste », « hallucine », « ahurissante », « bas…surprise », « sonnés », « retard », « étonnant ».

I-SURPRISE_POSITIVE (1,000) → Les termes annotés dans le fichier d'entrainement sont : « miracle », « étonné », « bonne nouvelle », « omg », « font une surprise », « chance », « waouh », « choqué », « incroyable », « ouf », « insolite », « heureusement », « incroyable ».

Dans la liste de termes porteurs de sentiments que nous avons utilisé (FEEL), nous avons remarqué que certains adjectifs ou verbes n'étaient pas répertoriés. Nous pouvons prendre l'exemple du verbe choquer et de l'adjectif choqué, dans la liste nous pouvons trouver « de manière choquant » mais pas les termes individuels.

Le terme « insolite » est trouvé dans le label I-SURPRISE_POSITIVE alors que dans la liste « insolite » possède le critère « surprise » avec la polarité négative. Il serait nécessaire que l'annotation soit raccord avec la liste ou inversement afin d'obtenir des résultats cohérents.

I-AMOUR (0,000) → Les termes annotés dans le fichier d'entrainement reflètent le sentiment d'amour avec sous des formes « d'expressions » ou d'ensembles qui font que l'amour est marquée mais implicitement. En effet, nous n'avons pas d'adjectifs explicites comme « amoureux » ou autre mais plutôt : « faire les yeux doux », « l'homme de ma vie », « me suis fait des amis », « t'épouse », « amour propre », « se ft d potes », « meilleurs amis », « s'embrassent », « beau fessier » (séquences relevées dans le fichier d'entrainement).

L'interprétation des résultats obtenus reste difficile.

6 Axes d'améliorations

6.1 Annotation

En parcourant notre corpus, nous pouvons nous apercevoir que certains éléments annotés ne correspondent pas à leur étiquette. Il semblerait qu'une annotation manuelle plus fine soit indispensable pour avoir un corpus de qualité et ainsi des résultats de qualité.

6.2 Enrichissement des traits

Comme vu précédemment dans l'évaluation, les résultats obtenus pourraient être meilleurs. Des perspectives d'améliorations sont donc envisageables. Les traits pourraient être enrichis pour donner un maximum d'informations pour l'étape d'apprentissage. Grâce à cet enrichissement nous pourrions obtenir des résultats plus satisfaisants car le système apprendrait plus d'informations qu'à présent. Aussi, nous pourrions essayer d'autres méthodes et d'autres classifieurs tels que SVM ou Naïve bayes pour observer les résultats obtenus et en conclure la méthode qui fonctionnerait le mieux avec nos traits.

6.3 Traitement plus rapide

Nous avons pu remarquer que l'exécution du programme requiert des ressources importantes (approximativement 45 minutes et jusqu'à 70% de la mémoire accordée à Spyder (IDE utilisé)). Dans l'optique d'optimiser notre programme, il pourrait être intéressant de repartir la puissance de calcul sur plusieurs cœurs (multithreading). Potentiellement, si l'on prend l'exemple de notre processeur composé de 4 cœurs (quad-core) et que chacun soit utilisé pour le calcul de notre programme de façon synchrone

et/ou asynchrone, le temps d'exécution pourra facilement être divisé par le nombre de cœurs voire d'avantages selon la méthode choisie.

7 Conclusion

Pour conclure ce travail de détection des opinions dans des tweets français, nous pouvons dire que c'est une tâche assez complexe. D'une part, parce qu'il faut que le format des données soit à l'identique du tutoriel. D'autre part, parce qu'il est compliqué de savoir comment améliorer les résultats au niveau de l'enrichissement des traits, ou de la modification de l'algorithme qui est assez compliqué à comprendre. De plus, l'apprentissage automatique est considéré, pour le linguiste en traitement automatique du langage, comme une « boîte noire » du fait que nous avons des résultats en sortie, mais nous ne pouvons pas observer comment le système à fait pour nous donner ces résultats. L'interprétation des résultats est difficile, par exemple, il est compliqué d'expliquer pourquoi le système nous donne un score de 0,000 pour la précision, le rappel et le F-mesure pour certains labels. Un autre inconvénient est le temps d'exécution du programme, il est difficile d'améliorer certains critères et d'observer les résultats rapidement car le programme s'exécute en approximativement 40 minutes.

Au niveau des résultats de l'évaluation, si nous prenons pour exemple les résultats obtenus par les participants du DeFT 2015 et du DeFT 2018, nos résultats semblent cohérents car presque similaires aux résultats qu'ils ont obtenus. L'utilisation de CRF semble être une bonne piste pour cette tâche, à condition que le script soit amélioré. Cependant, si nous nous basons sur les résultats récoltés lors du CoNLL-2002, nos résultats se trouvent en dessous des moyennes obtenues par les participants.

8 Bibliographie

Thierry Hamon, Amel Fraisse, Patrick Paroubek, Pierre Zweigenbaum, Cyril Grouin, (16 octobre 2017). *Analyse des émotions, sentiments et opinions exprimés dans les tweets : présentation et résultats de l'édition 2015 du défi fouille de texte (DEFT)*

Erik F. Tjong Kim Sang. *Introduction to the CoNLL-2002 Shared Task : Language-Independent Named Entity Recognition.*