Application de techniques de Data Mining en utilisant le logiciel RapidMiner

Ce laboratoire a pour objectif d'expérimenter un logiciel de Data Mining, à savoir RapidMiner. Celui-ci va nous permettre de mettre en place rapidement et simplement diverses tâches de prétraitement, d'appliquer des algorithmes de classification, de clustering, etc. et d'évaluer les résultats obtenus.

1 Classification de spams

Ci-dessous, la matrice de confusion résultant de la classification des SMS, avec le classificateur *Naive Bayes*.

		Prédiction	
		Spam	**Non-spam**
Vérité	**Spam**	92	3
	Non-spam	38	263

Table 1 – Classification des SMS avec un filtrage Bayesien.

Ce qui donne une accuracy de **89.65%**.

Dans le bloc "Process Documents from Data" nous n'avons pas mis d'étape de stemming. Est-ce que l'ajout de ce préprocessing a un impact sur les résultats obtenus ?

Une étape de Stemming Porter (algorithme adapté à la langue anglaise) a été ajoutée après le filtrage des stopwords. Ceci ne change quasiment pas les résultats obtenus, comme le montre la matrice de confusion suivante :

		Prédiction	
		Spam	**Non-spam**
Vérité	**Spam**	91	4
	Non-spam	38	263

Table 2 – Classification des SMS avec Naive Bayes et Stemming.

Cette nouvelle étape amène juste le système à se tromper sur un SMS en ne le classant pas comme spam, ajoutant donc un Faux Négatif.

Dans l'exemple ci-dessus, nous avons utilisé un classificateur bayésien, veuillez essayer d'autres familles de classificateurs, quel est l'impact sur le résultat obtenu ?

Tout d'abord, un *Arbre de décision* va avoir une excellente précision lorsqu'il va prédire un SMS comme étant un spam, il ne fait aucune erreur dans ce cas là. En revanche, il laisse passer beaucoup de spams en ne les détectant pas.

		Prédiction	
		Spam	**Non-spam**
Vérité	**Spam**	71	24
	Non-spam	0	301

TABLE 3 – Classification des SMS avec un arbre de décision.

Malgré tout, l'arbre de décision permet d'arriver à une accuracy de **93.94%**, ce qui est sensiblement meilleur qu'un filtrage bayésien.

Nous avons ensuite changé le classificateur pour un *Réseau de neurones*.

		Prédiction	
		Spam	**Non-spam**
Vérité	**Spam**	88	7
	Non-spam	4	297

TABLE 4 – Classification des SMS avec un réseau de neurones.

En ne commettant que 11 erreurs, sur 396 SMS, le réseau de neurones arrive à la meilleure accuracy parmis les classificateurs testés, à savoir **97.22%**.

Finalement, vous utiliserez la seconde source de données (emails.zip) sur laquelle vous appliquerez le même process. Que constatez-vous en comparant les 2 résultats ?

Nous avons au préalable eu à préprocesser les données des emails. Dans RapidMiner, nous avons ajouté un bloc permettant de transformer la colonne ńăspam ż en type binomiale et de lui définir le rôle de label (blocs *Numerical to Binomial* et *Set Role*).
De plus, le contenu de la colonneăavec le contenu des emails a été tronqué de " Subject : ". Ceci n'affecte évidemment pas le résultat, car il est présent dans chaque élément et n'est donc pas relevant et est ignoré par les algorithmes. Mais ce traitement permet de rendre le modèle réutilisable.

Avec un classificateur *Naive Bayes*, nous arrivons à la matrice de confusion suivante :

		Prédiction	
		Spam	**Non-spam**
Vérité	**Spam**	340	11
	Non-spam	38	1249

TABLE 5 – Classification des emails avec un filtrage Bayesien.

Avec une accuracy de **97.01%**, il s'agit d'un très bon résultat. Ajouter une étape de stemming n'améliore pas non plus le résultat sur ce type de données.

Nous avons également essayé d'utiliser un arbre de décision avec les emails. Celui-ci nous a permis d'arriver à une accuracy de **86.81%**, avec beaucoup de Faux Négatifs, ce qui est bien moins efficace. Un arbre de décision est de moins en moins efficace plus la taille et la complexité des données augmentent.

		Prédiction	
		Spam	**Non-spam**
Vérité	**Spam**	166	212
	Non-spam	4	1256

TABLE 6 – Classification des emails avec un arbre de décision.

2 Market basket analysis

2.1 Questions about association rules

Constatez-vous des différences dans les règles d'associations obtenues entre les 2 regroupements différents (par facture/par client) ? Veuillez commenter vos résultats. Est-il possible de générer une/des autre/s colonne/s à partir des donnees initiales qui produisent des règles interessantes ?

Pour la création des fichiers CSV utilisés pour la génération de règles d'associations, un nettoyage des données a été fait.
Les lignes avec des données vides, comme par exemple le nom du produit sont filtrées. Il en est de même si la quantité est plus petite que 0. Un programme a été implémenté afin de détecter et filtrer les lignes avec des éventuels noms de produit incohérents.

Voici une liste des noms incohérents trouvés : *wet pallet,sold in set ?,wet damaged,faulty,posy candy bag,michel oops,wet/mouldy, lost,dagamed,daisy notebook,fba,taig adjust,water damage,ribbons purse, ?display ?,found in w/hse,damaged stock, samples, mixed up, ???missing,breakages,chilli lights,dotcom set,re-adjustment, packing charge,amazon sales, ?, mix up with c,owl doorstop,stock check,mia,wrap folk art,carriage,damages, ?lost,john lewis,found box,20713,show samples,damages wax,website fixed,jumbo bag owls,wet boxes,frog candle,found,given away,mouldy, ???,broken, wrong code,damages ?,display,wet,bunny egg box,polkadot pen, ?missing,dotcom,amazon adjust,damaged,returned, toybox wrap,smashed, ?sold as sets ?,showroom,led tea lights,wet/rusty,missing ?,wrong barcode,lost in space, crushed ctn,popcorn holder,dotcom sales,lost ??,sold as 1,bingo set,thrown away.,crushed boxes,test,check,crushed, wet ?,mailout,amazon,rain poncho,adjust,ebay,adjustment,spotty bunting,wet rusty, cracked, sale error, ???lost, water damaged,missing,garage key fob,dotcomstock,dotcom adjust,sold as 22467, ??,dotcom postage,jumbo bag toys, shoe shine box, ?? missing,on cargo order,wrong code ?,check ?,label mix up,postage, wrap carousel,can't find,wrongly marked,retrospot lamp,thrown away,counted, ????missing,cordial jug*

De plus, chaque donnée de colonne est systématiquement trimée pour éviter tout problème et pour faciliter la suite du traitement. Voici quelques statistiques calculées lors du filtrage :

- Nombre de lignes dans le fichier : 541'909
- Nombre de lignes après nettoyage : 397'924
- Nombre de produits : 3'639

Le programme a donc retiré 143'985 lignes.

Un autre programme a été implémenté pour standardiser la description des produits en la liant avec le numéro du produit et celle qui est la plus utilisée, dans le cas un produit aurait plusieurs descriptions.
Pour voir une partie de l'algorithme, il faut regarder la fonction *Main.resolveProductDescription()* dans le code Java fourni en annexe.

Ci-dessous, d'autres informations que nous calculons lors de la création des nouveaux fichiers CSV.

Création du fichier par pays (permets de tester rapidement le code) :
- Nombre de lignes dans le nouveau fichier byCountry.csv : 37
- Durée du traitement : 00 :01.625

Création du fichier par client :
- Nombre de lignes dans le nouveau fichier byCustomer.csv : 4'339
- Durée du traitement : 01 :44.361

Création du fichier par facture :
- Nombre de lignes dans le nouveau fichier byInvoice.csv : 18'536
- Durée du traitement : 07 :26.901

Afin de pouvoir générer ces données, nous avons utilisé du multi-threading ce qui nous a permis d'améliorer considérablement le temps nécessaire à la création des fichiers.

Pour lire le fichier, nous avons dû en créer un nouveau et copiant et collant son contenu. Nous pensons qu'il

doit y avoir un problème d'encodage.

Pour tester le programme, il suffit de lancer la commande *mvn clean install*, de modifier le fichier *application.properties* avec l'emplacement des fichiers de base et de destination.

La création de ces fichiers n'est pas simple et nous a pris beaucoup de temps. Par exemple, nous pourrions encore améliorer la génération, en tenant compte des produits retournés. Il serait aussi bien d'avoir un meilleur descriptif de ces données.

3 Utilisation d'un WordNet sur des commentaires dutilisateurs

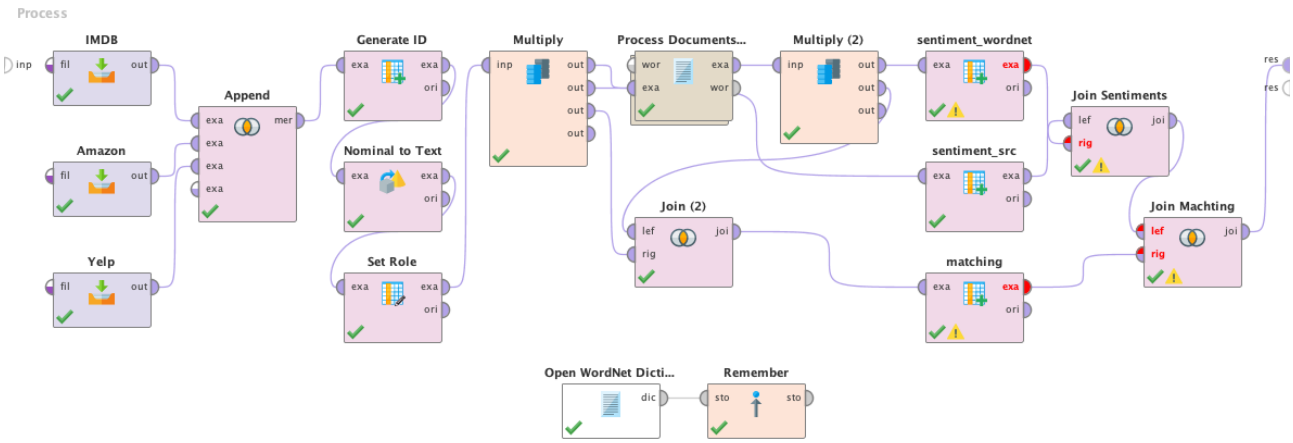


FIGURE 1 – Processus complet : WordNet sur 3000 commentaires

Dans le cadre de cette 3e partie, nous avons mis en place un processus capable de comparer des commentaires sous format texte fourni avec un étiquetage du sentiment (Positif, Négatif, et Neutre), avec un algorithme de prédiction capable d'extraire le sentiment d'un texte basé sur le dictionnaire WordNet. Pour simplifier la comparaison, nous avons ajouté un attribue qui vérifie que le sentiment source est le même que le sentiment prédit (appelé chez nous matching).

Nous allons pour ce faire prétraiter le texte fourni à l'aide de combinaisons de différentes techniques et fournir le résultat au module "Extract Sentiment de WordNet" qui générera une prédiction du sentiment lié à l'input.

Nous utiliserons les modules suivants :
- tokenisation
- transformation des caractères
- stopwords
- filtre sur la longueur des tokens
- stemming

3.1 Questions sur les règles d'association

Commentez les resultats obtenus par rapport aux etiquettes existantes sur le dataset. Quelle est l'influence des différentes étapes de "text processing" sur le résultat que vous obtenez ?

3.1.1 Module Extract Sentiment WordNet par défault

Dans le but d'explorer les influences lié aux différentes étapes de "text processing" nous allons utiliser le module de WordNet qui permet d'extraire le sentiment d'input sous format texte avec ses paramètres par défaut. (Fig.2) (Nous verrons plus tard que la notion de texte sera faite sous forme de tokens, qui est une décomposition d'un texte en mots.)

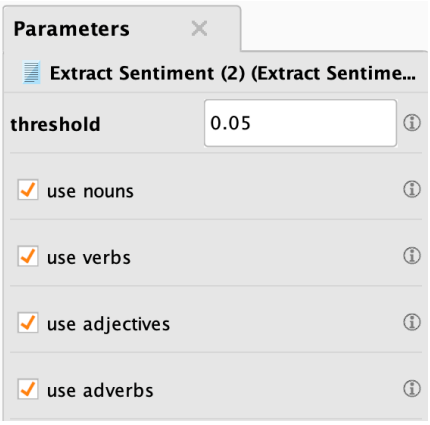


FIGURE 2 – Extract Sentiment Default Threshold at 0.05

3.1.2 Process Documents from Data uniquement avec le module d'extraction du sentiment WordNet

Nous constatons un matching à **0%**. En effet, le module "Extract Sentiment" n'est pas capable d'interpréter le texte brut des commentaires et par conséquent donne une sortie commune Neutre (basé sur la valeur numérique 0) à toutes les entrées.

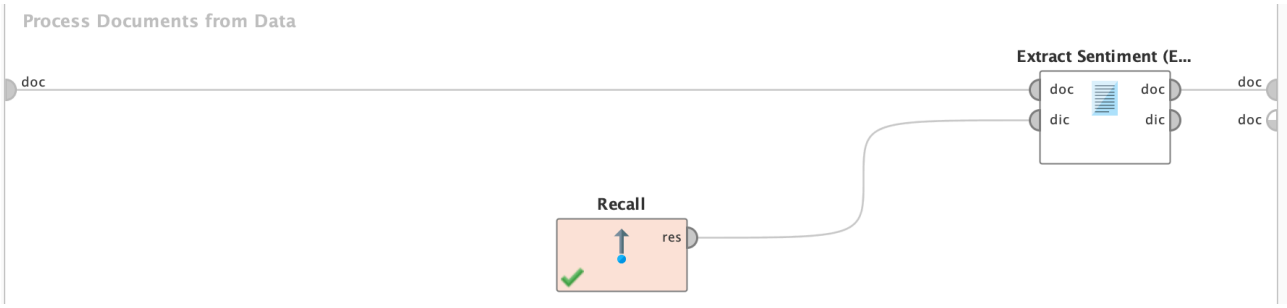


FIGURE 3 – Process Documents from Data with only the sentiment extractor from WordNet

Row No.	id	sentiment_...	sentiment_...	matching
1	id_1	Negative	Neutral	false
2	id_2	Negative	Neutral	false
3	id_3	Negative	Neutral	false

matching
Nominal
Missing: 0
Least: false (3000)
Most: false (3000)

FIGURE 4 – Results of the Process Documents from Data with only the sentiment extractor from WordNet

3.1.3 Process Documents from Data composé de la tokenisation

En utilisant la tokenisation avec son paramètre par défaut (non letters) le module "Extract Sentiment" a maintenant les ressources minimums pour fonctionner et permet une prédiction à **29.7%**. (Fig.6)
Il est intéressant de noter qu'il y a plusieurs options pour la tokenisation. En jouant avec ces paramètres, nous

obtenons les résultats suivants : (notons : que nous cherchons à augmenter le nombre de matchs (true) de l'attribue matching)

- non letters (par défaut) : **70.3%** (Fig.6)
- special characters (. :) : **67.3%**
- regular expression : Pas essayé
- linguistic sentences : **0%**
- linguistic tokens : **69.8%**

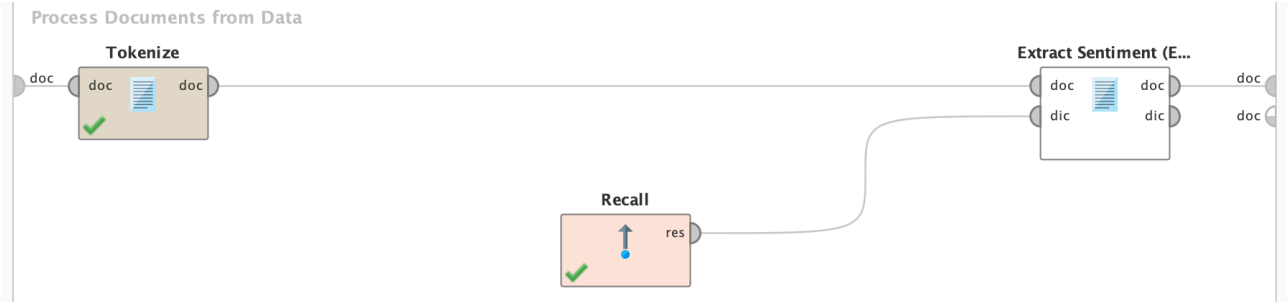


FIGURE 5 – Process Documents from Data with tokenization

Row No.	id	sentiment_...	sentiment_...	matching
1	id_1	Negative	Positive	false
2	id_2	Negative	Negative	true
3	id_3	Negative	Negative	true

matching
Nominal
Missing: 0
Least: false (892)
Most: true (2108)

FIGURE 6 – Results of tokenization via non letters

3.1.4 Process Documents from Data composé de la tokenisation ainsi que de la transformation en caractères minuscules

Nous ajoutons la transformation des tokens, en utilisant le paramètre par défaut : lower case. Nous avons également essayé la seconde option : upper case, cependant le résultat est le même. Ce module uniformise les lettres des tokens en soit minuscule ou minuscules. Ce traitement nous à fait perdre un match, cependant nous gardons notre **70.3%**. Il est intéressant de constater que pour certains commentaires, l'effet de la capitalisation irrégulière puisse avoir un impact.

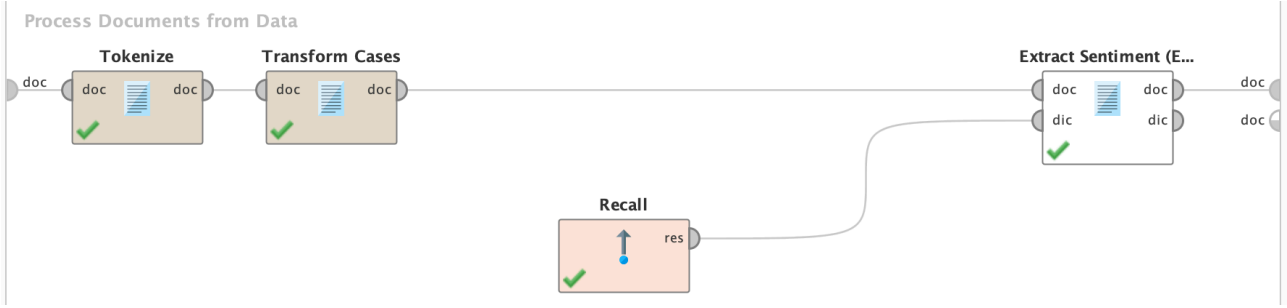


FIGURE 7 – Process Documents from Data with tokenization and low character filter

Row No.	id	sentiment_...	sentiment_...	matching
1	id_1	Negative	Positive	false
2	id_2	Negative	Negative	true
3	id_3	Negative	Negative	true

matching
Nominal
Missing: 0
Least: false (890)
Most: true (2110)

FIGURE 8 – Results of the Process Documents from Data with tokenization and low character filter

3.1.5 Process Documents from Data composé de la tokenisation, la transformation en caractères minuscules, et la suppression des stopwords

En ajoutant le filtre des stopwords anglais. Nous constatons une perte en performance des prédictions avec **66.2%**. Ce n'est pas le comportement que nous attendons généralement d'un filtre par stopword, au contraire. Nous pensons que cet effet est dû à la taille du corpus et de la perte du contexte lorsqu'il y a un sentiment fort, où l'utilisateur a tendance à accumuler via des mots de liaisons (inclue dans les stopwords) des adjectifs pour accentuer une opinion.

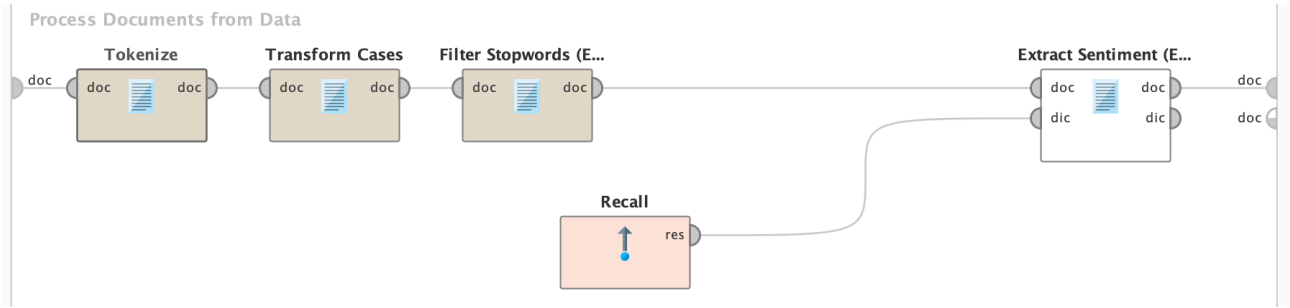


FIGURE 9 – Process Documents from Data with the tokenizion, the low cases, and stopword filter

Row No.	id	sentiment_...	sentiment_...	matching
1	id_1	Negative	Negative	true
2	id_2	Negative	Negative	true
3	id_3	Negative	Negative	true

matching
Nominal
Missing: 0
Least: false (1013)
Most: true (1987)

FIGURE 10 – Results on the Process Documents from Data with the tokenizion, the low cases, and stopword filter

3.1.6 Process Documents from Data composé de la tokenisation, la transformation en caractères minuscules, la suppression des stopwords, et un filtre sur la longueur des tokens

En utilisant les paramètres par défauts (Fig.13a) nous obtenons une performance de **60.9%**. Cependant en tweakant les paramètres (1 en taille minimum et en gardant la taille maximum par défault 14)(Fig.13b) nous arrivons à atteindre une performance de **66.3%**.(Fig.14)

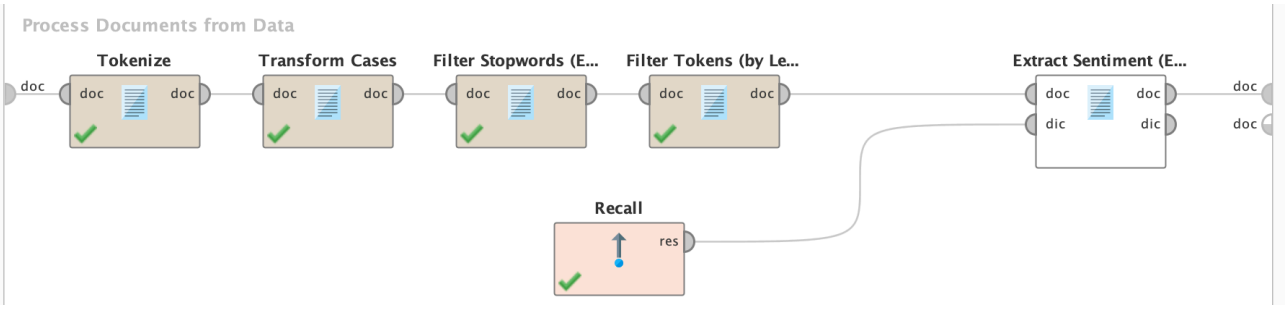
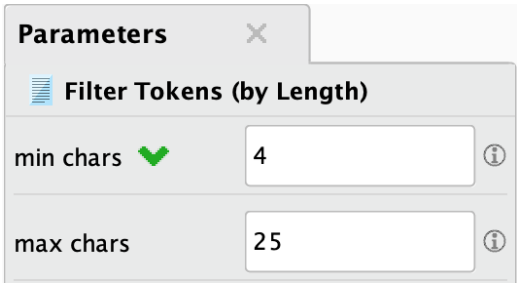


FIGURE 11 – Process Documents from Data with all elements expect stemming

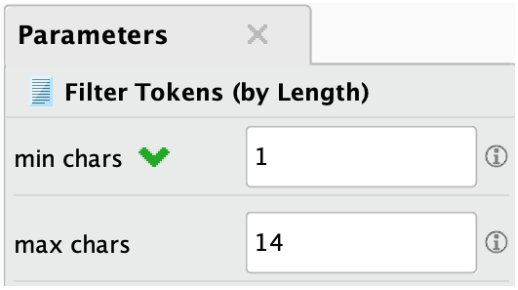
Row No.	id	sentiment_...	sentiment_...	matching
1	id_1	Negative	Negative	true
2	id_2	Negative	Negative	true
3	id_3	Negative	Negative	true

matching
Nominal
Missing: 0
Least: false (1172)
Most: true (1828)

FIGURE 12 – Results for stemming with the defaults parameters



(a) Default parameters from the length filter



(b) Best parameters from the length filter

FIGURE 13 – Parameters from the length filter

Row No.	id	sentiment_...	sentiment_...	matching
1	id_1	Negative	Negative	true
2	id_2	Negative	Negative	true
3	id_3	Negative	Negative	true

matching
Nominal
Missing: 0
Least: false (1011)
Most: true (1989)

FIGURE 14 – Results of the best parameters from the length filter

3.1.7 Process Documents from Data en ajoutant le stemming

Voir figure 15 pour la topologie. Nous avons essayé les différents modules de stemming à disposition et avons obtenu des résultats différents pour chacun :

— Snowball : **57.9%**
— Porter : **56.5%**
— Lovins : **52.8%**
— WordNet : **68.2%**(Fig.17) avec les paramètres par défauts (Fig.16a)

Nous basant sur ces résultats, nous avons sélectionné le stemming via le WordNet et avons ajouté l'option pour autoriser les ambiguïtés (Fig.16b), ce qui nous à permis d'obtenir un petit gain de performance pour monter à **68.4%**.

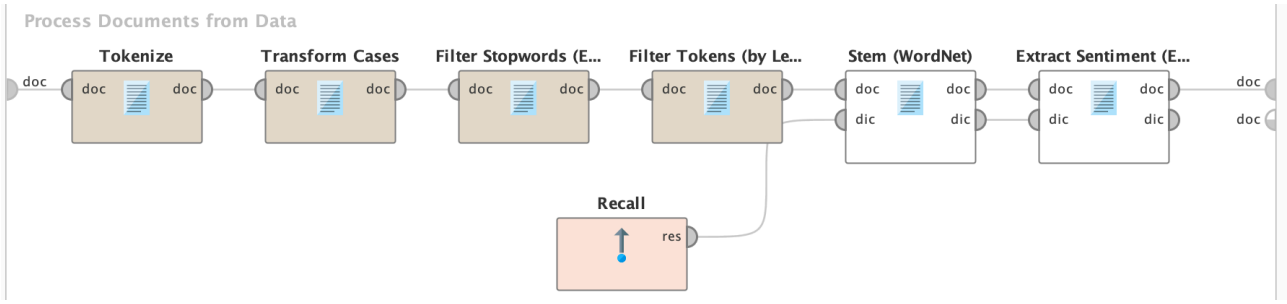
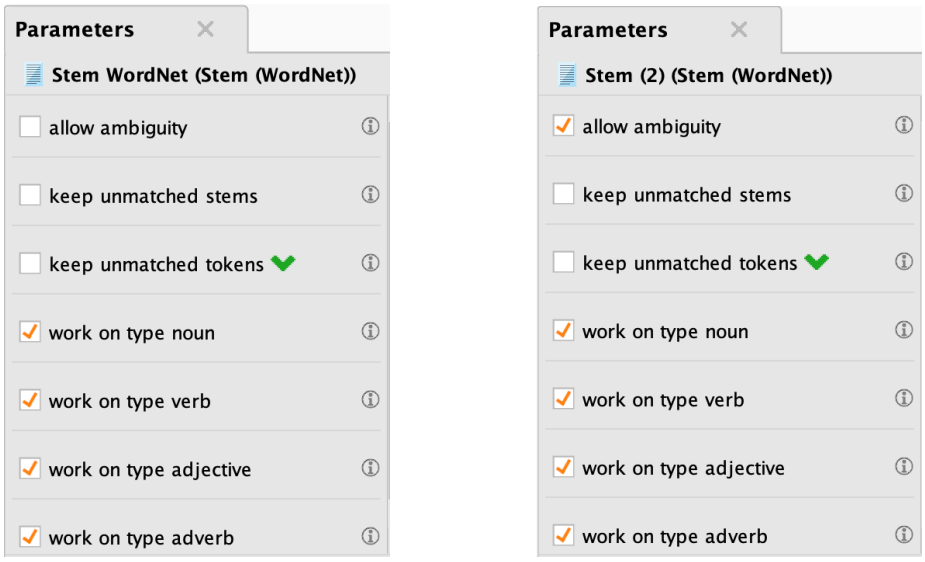


FIGURE 15 – Process Documents from Data with all elements



(a) Default parameters from the stem via WordNet

(b) Best parameters from the stem via WordNet

FIGURE 16 – Parameters from the stemming via WordNet

Row No.	id	sentiment_...	sentiment_...	matching
1	id_1	Negative	Negative	true
2	id_2	Negative	Negative	true
3	id_3	Negative	Negative	true

matching
Nominal
Missing: 0
Least: false (955)
Most: true (2045)

FIGURE 17 – Results on using the default WordNet stemming parameters

Row No.	id	sentiment_...	sentiment_...	matching
1	id_1	Negative	Negative	true
2	id_2	Negative	Negative	true
3	id_3	Negative	Negative	true

matching
Nominal
Missing: 0
Least: false (948)
Most: true (2052)

FIGURE 18 – Results on using the best WordNet stemming parameters

3.1.8 Modification du module d'extraction de sentiment

Il nous reste, à présent, plus que le module d'extraction de sentiment à customiser. En effet, jusqu'à présent nous avons utilisé sa configuration par défaut (Fig.2), qui nous a permis d'obtenir performance cumulée de **68.4%** (Fig.16b). En jouant avec les paramètres à disposition(Fig.19) nous avons réussit à obtenir une performance de **70.4%**.(Fig.20)

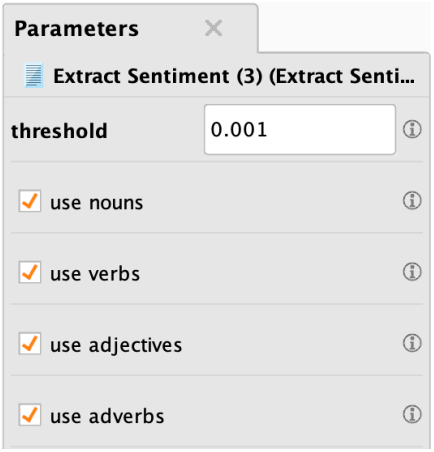


FIGURE 19 – Extract Sentiment with threshold at 0.001

Row No.	id	sentiment_...	sentiment_...	matching
1	id_1	Negative	Negative	true
2	id_2	Negative	Negative	true
3	id_3	Negative	Negative	true

matching
Nominal
Missing: 0
Least: false (889)
Most: true (2111)

FIGURE 20 – Results on using the best WordNet stemming parameters

3.2 Notre meilleur modèle de prédiction cumulée

Nous n'avions qu'un module nous dégradant la prédiction(Fig.9), le filtre de stopwords. Nous l'avons donc supprimée de notre topologie (Fig.21) et terminons avec une performance de **72.6%**.(Fig.22)

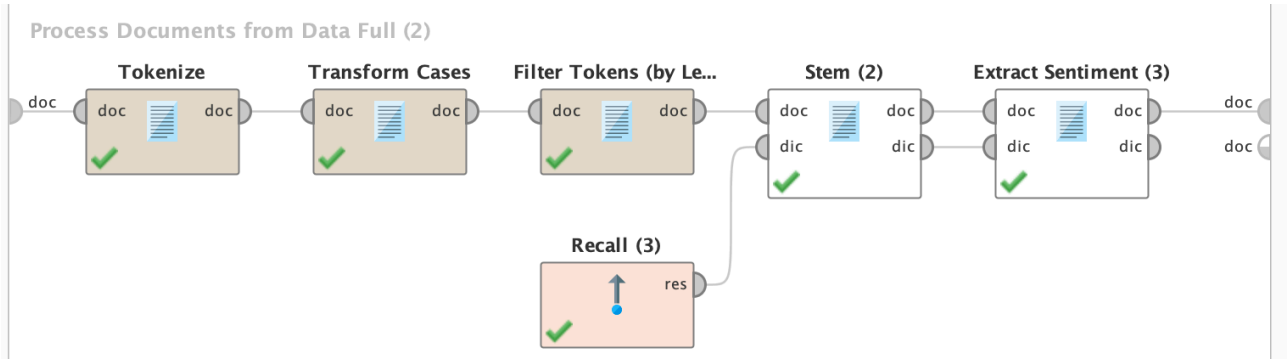


FIGURE 21 – Process Documents from Data with optimized elements

Row No.	id	sentiment_...	sentiment_...	matching
1	id_1	Negative	Negative	true
2	id_2	Negative	Negative	true
3	id_3	Negative	Negative	true

matching
Nominal
Missing: 0
Least: false (822)
Most: true (2178)

FIGURE 22 – Results from our best model

4 Conclusion