Analyse sentimentale à grande échelle avec PySpark

DELTEIL Clément - DELC12110004

SIRVENT Thomas - SIRT10080007

1.
Introduction

NLP & Rappel de l'article

Natural Language Processing

Utile…

- Connaître les opinions
- Identifier les intérêts

mais difficile…

- Les machines ne comprennent pas le texte
- Le langage humain est complexe à comprendre

Rappel de l'article

Techniques utilisées

- Bloom Filter
- kNN parallélisé avec MapReduce

Nodarakis, Nikolaos et al. “Large Scale Sentiment Analysis on Twitter with Spark.” EDBT/ICDT Workshops (2016).

Map

Calcul des distances

Shuffle

Regroupement des paires <classe, distance>

Reduce

Choisit les "k" plus proche voisins et fait un vote majoritaire

2.

Logistic Regression

Implantation dans Spark

Descente de Gradient

Descente de Gradient par lots

- Modèle mis à jour avec toutes les données

$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta)$$

Descente de Gradient Stochastique

- Estimation de la descente de gradient par lots. La taille du lot est égale à 1

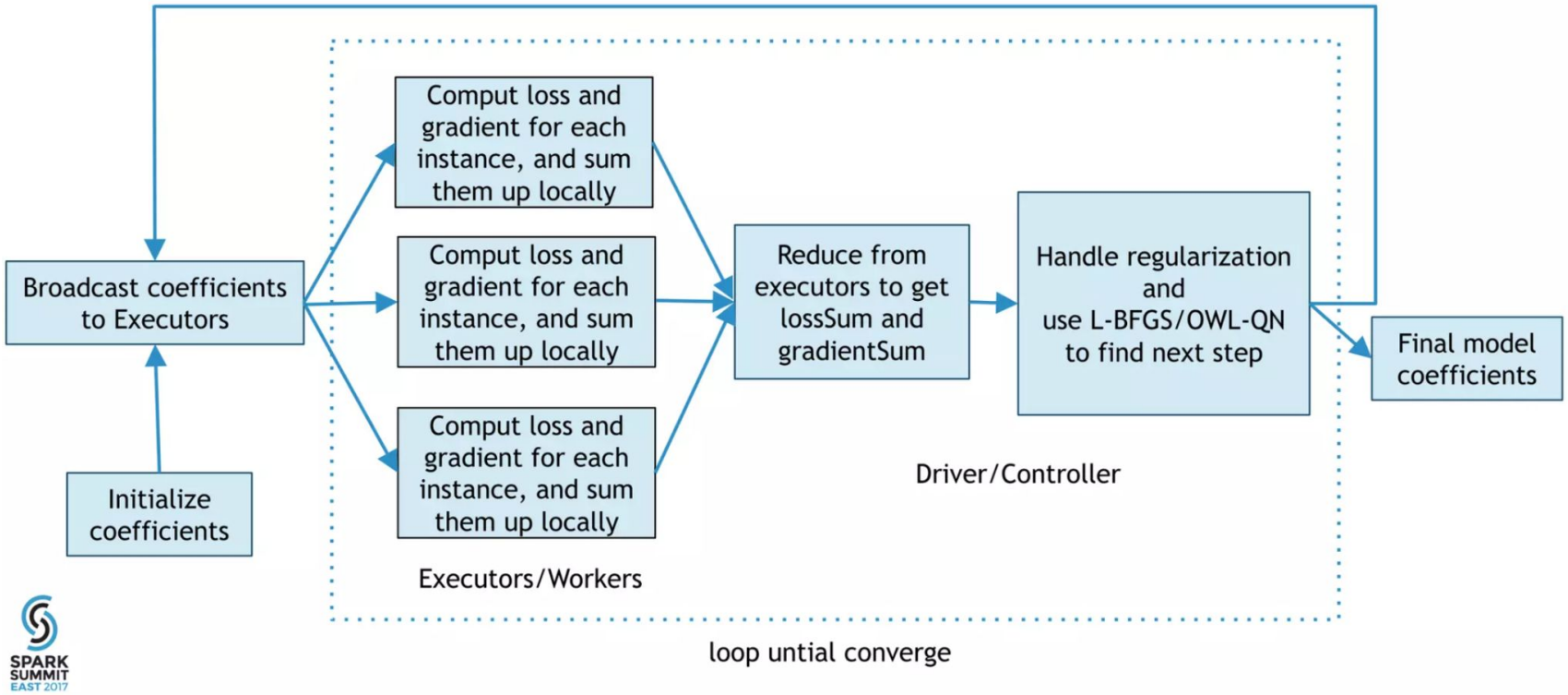
$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta; x^{(i)}; y^{(i)})$$

Descente de Gradient par mini-lots

- Généralisation de la descente de gradient stochastique.
- Taille du lot = n

$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta; x^{(i:i+n)}; y^{(i:i+n)})$$

Implantation



3.

Jeux de données

Sentiment140 & Custom

Sentiment140 & Custom

Sentiment140

1 600 000 Tweets (50/50 positifs négatifs)

Champs :

- Target
- Tweet

Données déjà nettoyées

Traitement des données

Mots vides, ponctuation, #hashtag, @Username

Custom

600 000 Tweets récupérés

Requête :

("elon musk" OR "@elonmusk" OR "elonmusk") -is:retweet lang:en

4.

Extraction des caractéristiques

Tokenizer, HashingTF, CountVectorizer, TF-IDF, N-Gram, ChisQSelector

Différents essais via Pipeline

Entrée [9]:

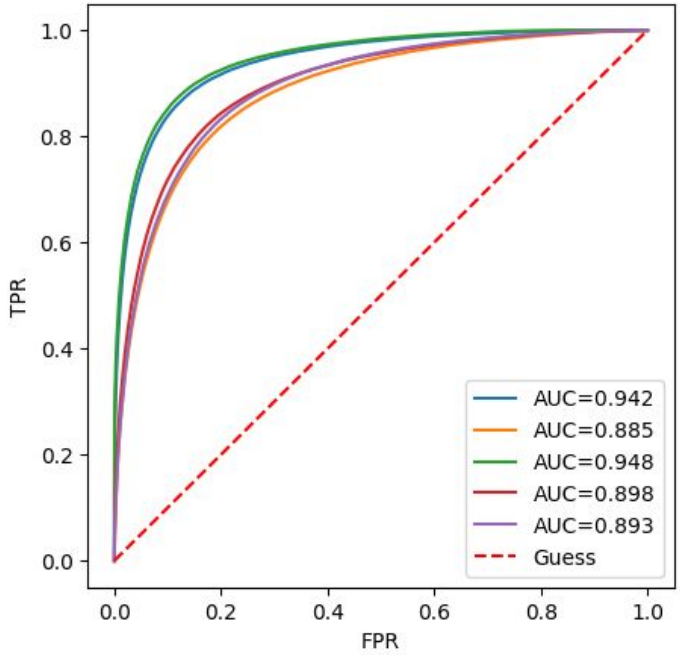
```
tokenizer = Tokenizer(inputCol="tweet", outputCol="words")
hashtf = HashingTF(inputCol="words", outputCol='tf')
idf = IDF(inputCol='tf', outputCol="features")
label_stringIdx = StringIndexer(inputCol = "target", outputCol = "label")
lr = LogisticRegression()
evaluator = MulticlassClassificationEvaluator(predictionCol="prediction")
pipeline = Pipeline(stages=[tokenizer, hashtf, idf, label_stringIdx, lr])
```

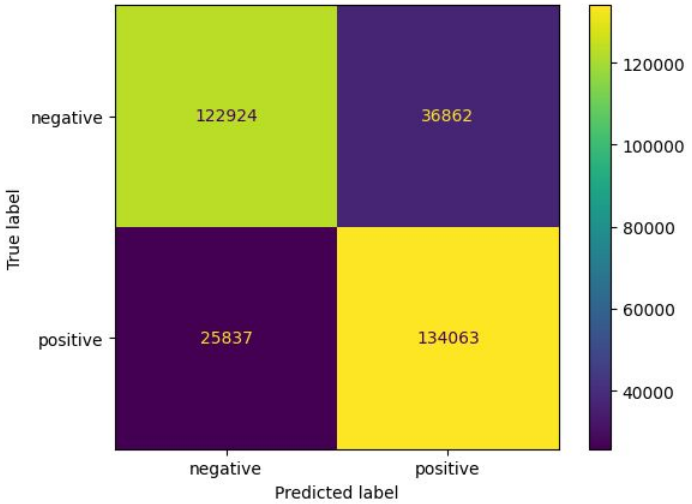
Entrée [10]:

```
tokenizer = Tokenizer(inputCol="tweet", outputCol="words")
cv = CountVectorizer(vocabSize=2**16, inputCol="words", outputCol='cv')
idf = IDF(inputCol='cv', outputCol="features", minDocFreq=5) #minDocFreq: remove sparse terms
label_stringIdx = StringIndexer(inputCol = "target", outputCol = "label")
lr = LogisticRegression()
evaluator = MulticlassClassificationEvaluator(predictionCol="prediction")
pipeline = Pipeline(stages=[tokenizer, cv, idf, label_stringIdx, lr])
```

49 152 features → 16 384 features

Matrice de confusion et courbes AUC





5.
Résultats

Sélection des meilleures caractéristiques et modèles

Tableaux récapitulatifs

Features	Logistic Regression	Naive Bayes	SVM
Hashing TF-IDF + 1-Gram	73.4	72.5	75.8
Hashing TF-IDF + 1-Gram	77.7	74.9	77.8
CountVectorizer TF-IDF + 1-Gram	76.5	75.8	78.3
CountVectorizer TF-IDF + 1-Gram	79.3	76.8	79.5
CountVectorizer TF-IDF + 1-2-3-Gram + ChisQSelector	80.8	78.7	80.4

TABLE 1 – Précision des modèles dans chaque scénario

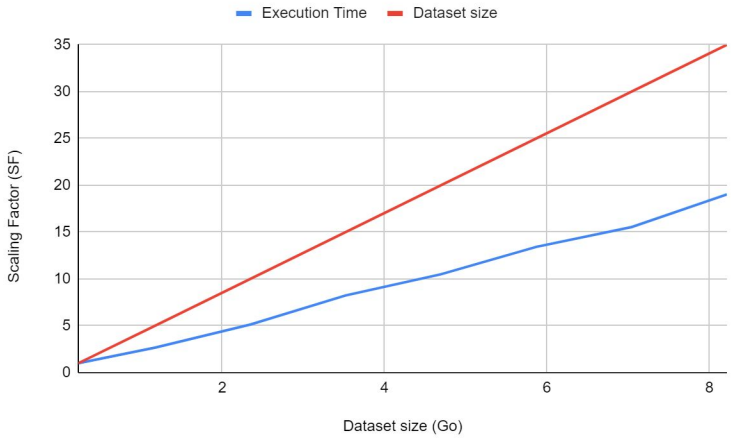
Metrics	Logistic Regression	Naive Bayes	SVM
False Negative	25 837	37 465	37 600
False Positive	36 862	30 725	24 934
True Negative	122 924	129 061	122 186
True Positive	134 063	122 435	134 966
Accuracy	0.808	0.787	0.804
Precision	0.809	0.787	0.806
Recall	0.808	0.787	0.804
F1-Score	0.808	0.787	0.805

TABLE 2 – Tableau récapitulatif

6.
Passage à l'échelle

Google Cloud Cluster

Cluster Google Cloud



Configuration

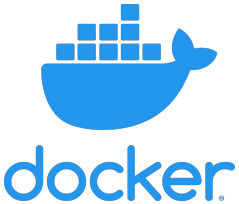
- **Master** :

  4 processeurs virtuels, 16 Go de RAM

- **Esclaves** :

  6 * 4 processeurs virtuels, 15 Go de RAM

	Nom	Rôle
✓	sparkboi-m	Maître
✓	sparkboi-w-0	Nœud de calcul
✓	sparkboi-w-1	Nœud de calcul
✓	sparkboi-w-2	Nœud de calcul
✓	sparkboi-w-3	Nœud de calcul
✓	sparkboi-w-4	Nœud de calcul
✓	sparkboi-w-5	Nœud de calcul

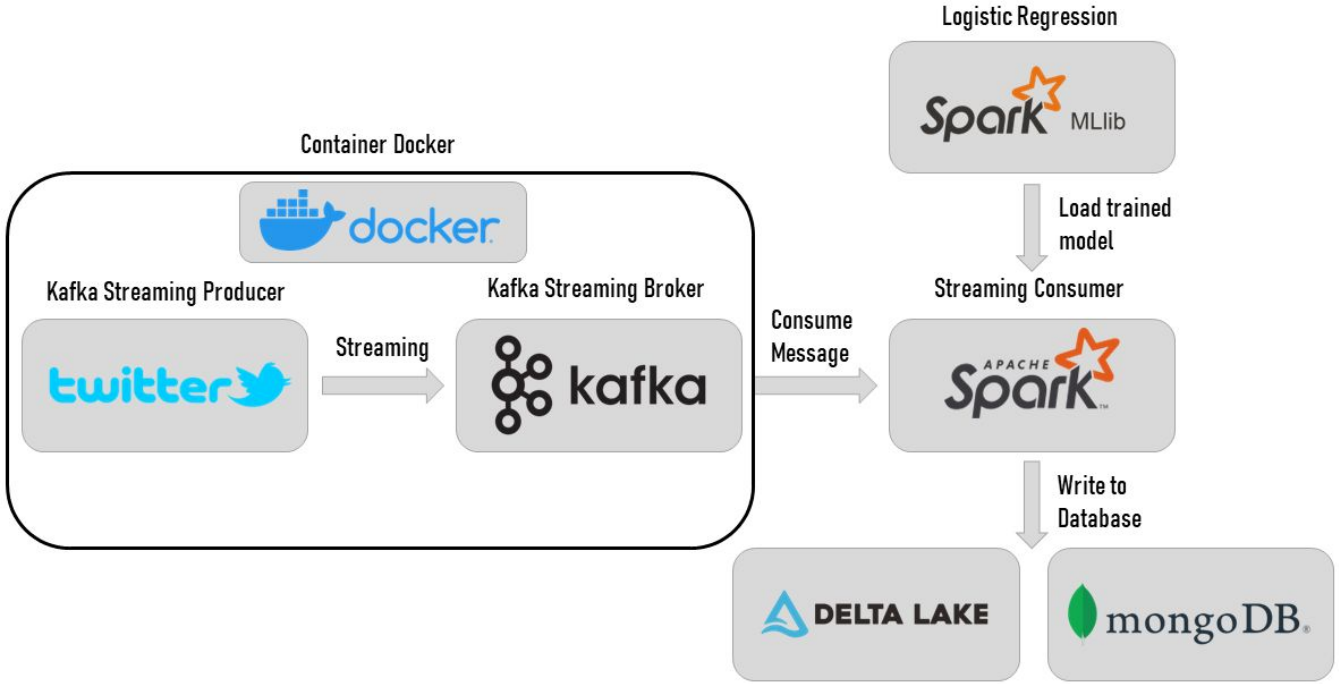


7.

Pour aller plus loin...

ETL, Spark Streaming, Kafka, Docker

ETL Pipeline (Spark Streaming)



Prédictions en temps-réel

```
Batch: 2
-------------------------------------------
+--------------------+----------+
|        cleaned_data|prediction|
+--------------------+----------+
|[litmormon, zeroh...|       1.0|
|[work, careful, t...|       1.0|
|[closer, with, pa...|       0.0|
|[ctg, is, hiring,...|       1.0|
|[is, there, a, wa...|       0.0|
|[improve, push, s...|       0.0|
|[radio, bit, note...|       0.0|
|[return, police, ...|       0.0|
|[there, are, many...|       1.0|
```

8. Conclusion

Merci !

Des questions ?

Clément Delteil & Thomas Sirvent

Credits

Special thanks to all the people who made and released these awesome resources for free:

- Presentation template by SlidesCarnival
- Photographs by Unsplash