HAL open science

Besoins et avantages de la fouille de données textuelles en sciences agronomiques

Ines Abdeljaoued-Tej

To cite this version:

Ines Abdeljaoued-Tej. Besoins et avantages de la fouille de données textuelles en sciences agronomiques. 2020. hal-02126728v2

HAL Id: hal-02126728

https://hal.science/hal-02126728v2

Preprint submitted on 4 Jan 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Besoins et avantages de la fouille de données textuelles en sciences agronomiques

Ines Abdeljaoued-Tej

Laboratoire BIMS, LR16IPT09, Institut Pasteur de Tunis, Université Tunis El Manar
ESSAI, Université de Carthage, Tunisie.
inestej@gmail.com

Abstract

Farming systems include intensive techniques, No-tillage, minimum intervention or organic agriculture. These systems require farmers to have a precise knowledge of agricultural practices. To this, we can add the expertise of new technologies, the control of treatment resistance, the gain of knowledge on seed varieties or the impact on the soil. This mass of information is available on the net : in scientific articles, discussion forums, specialized websites, and social networks. It is information in text format, usually unstructured. The objective of this work is to provide an overview of research on textual data mining in agriculture. We present the main methods for extracting relevant information. We test the data mining on data from Scopus, Twitter and a commercial site for agricultural products. We give an example of data classification via machine learning tools. The code to perform this review is on Python.

Keywords: Machine learning, sentiment analysis, web scraping, smart farming.

Introduction

Le traitement du langage naturel est un des domaines les plus en vue de l'intelligence artificielle. Il s'agit principalement de la parole et du texte qui sont produits, à des quantités faramineuses sous forme d'emails, de SMS, de pages web : blogs, réseaux sociaux. Le langage naturel est la façon dont nous, humains, communiquons les uns avec les autres. Si nous observons les divers domaines utilisant aussi bien le langage naturel que les ressources techniques les plus modernes, nous constatons aisément que l'agriculture, et plus particulièrement les sciences agronomiques, sont actuellement révolutionnées par l'essor du numérique.

L'agriculture de précision ou le « smart farming » permet d'aider à la décision à partir de données locales (Pivoto, Waquil, Talamini, Finocchio, Corte, and de Vargas Mores 2018). L'état des plantes, la qualité de la terre mais aussi l'interrogation des sources de données distantes (prévision météo, archives, tweets) permettent d'aider les agriculteurs à suivre et à prévoir l'évolution des cultures. Dans (Chebrolu, Lottes, Schaefer, Winterhalter, Burgard, and Stachniss 2017) un robot photographie régulièrement des plants de betterave et collecte des informations locales. Ce sont des données qui, utilisant des techniques d'apprentissage automatique (principalement des outils statistiques et des réseaux neuronaux), permettent d'aider au suivi du risque de maladies, du développement des cultures, etc. Ainsi, cette démarche pourrait être appliquée en commandant des automates à distance (irrigation, robots de traite, robots d'alimentation). La décision peut aussi être diffusée vers des systèmes distants comme le lancement d'alertes via des SMS ou des emails (évènement climatique, procédure de traitement lors des infections). Dans (Da Costa, Tjandrasa, and Djanali 2018) ce sont les différentes maladies du riz et les traitements préconisés qui sont retenus et étudiés. Les effets secondaires et l'évaluation de traitements sont obtenus grâce à un traitement de la langue sur des corpus spécifiques.

L'essor du monde agricole et la modernisation des techniques sont un terreau pour les nouvelles méthodes de fouille textuelle et d'intelligence artificielle, qui trouvent aujourd'hui leur application en agriculture numérique. De plus, l'étude du comportement des techniciens ou des exploitants agricoles permet de reproduire des modèles, grâce à l'intelligence artificielle, où le comportement est une valeur monnayable (Măruşter, Faber, Jorna, and Van Haren 2008). Sachant l'importance du traitement des données textuelles, et plus particulièrement, l'extraction de connaissances à partir de textes (blogs, forums de discussion), la fouille de données textuelles apporte déjà des résultats intéressants (Liu, Chen, Dai, Guo, Zhang, and Chen 2018). Dans (Comer 2016), c'est l'étude du statut des femmes agricultrices qui est étudié, via une revue des articles de presse. Ce traitement médiatique de la parité, pourrait être généralisé à d'autres sources de données et obtenu de manière systématique, via la fouille automatique. Par exemple, l'analyse des revues littéraires (Zschocke 2019) est désormais chose courante, puisque les données sont à portée de main (voir Scopus par exemple de Elsevier Burnham (2006)).

Nous nous intéressons particulièrement à des données textuelles, souvent mal structurées, liées à l'agriculture. Le corpus inhérent à cette thématique est aussi bien technique que pratique. Parmi les études déjà menées, nous observons la fouille automatique d'articles scientifiques portant sur l'efficacité des traitements administrés aux plantes (Ricroch, Harwood, Svobodová, Sági, Hundleby, Badea, Rosca, Cruz, Salema Fevereiro, Marfà Riera, Jansson, Morandini, Bojinov, Cetiner, Custers, Schrader, Jacobsen, Martin-Laffon, Boisron, and Kuntz 2016). Ce sont les nouveaux challenges dans la recherche en agronomie, comme le montre le nombre croissant d'articles scientifiques sur le sujet. Un autre volet porte sur l'extraction d'informations à partir de données expérimentales du domaine agricole : c'est une approche plus technique puisqu'elle aborde des questions d'ingénierie de la connaissance issues de données expérimentales extraites d'articles scientifiques, afin de pouvoir les intégrer dans des systèmes d'aide à la décision. Dans (Devyatkin, Nechaeva, Suvorov, and Tikhomirov 2018) nous pouvons voir quels secteurs sont les plus touchés par la recherche (une étude sur les résumés d'articles publiés entre 2008 et 2015, où le plus gros des thèmes examinés porte sur la sélection de semences). Tout cela repose sur le génie des connaissances et des systèmes (KSE [1]), sur le traitement du langage naturel (NLP [2]) ainsi que sur des approches d'exploration de données guidées par des ressources ontologiques et terminologiques.

Les projets qui visent à fournir de nouveaux outils décisionnels efficaces aident le développement agricole ainsi que la protection de la bio-diversité (Basnet and Bang 2018). Plus spécifiquement, il s'agit d'acquérir de nouvelles données, d'échanger des données, d'extraire des connaissances et de communiquer des informations utiles entre différents acteurs impliqués dans les divers domaines de l'agriculture. Les algorithmes utilisés sont les machines à vecteurs de support (SVM), les réseaux de neurones (NN), les forêts aléatoires (RF) et l'algorithme Naive Bayes (NB). Nous avons opté pour le logiciel Python (Rossum 1995; Sanner et al. 1999). En effet, des travaux comme (Bird, Klein, and Loper 2009) utilisent Python pour le traitement automatique de texte. D'autres logiciels de fouille de données existent comme Weka (Garner et al. 1995) ou R (R Core Team 2017), qui contient des librairies spécifiques aux données textuelles (Silge and Robinson 2017).

Dans ce travail, nous effectuons une revue des outils couramment utilisés dans la fouille textuelle en agriculture. Le reste de ce document est détaillé comme suit : la section 1 montre comment se fait la collecte des données textuelles. Dans la section 1.1 nous détaillons les solutions utilisées. Nous nous concentrons sur les réseaux sociaux et les sites de commerce électronique (ou *e-commerce*). La section 1.2 montre la pratique de traitement préliminaire des données. La section 1.3 indique comment sont représentées les données : la conversion des mots en vecteurs, et les vecteurs en matrices par exemple. La section 2 explore l'analyse et la fouille des données textuelles via l'analyse sentimentale. La section 2.1 donne un exemple partant de la collecte à l'analyse des données textuelles, puis la section 2.2 détaille

1. Knowledge and Systems Engineering
2. Natural Language Processing

la classification des données textuelles d'un site de e-commerce. Les codes utilisés dans nos exemples ont été ajoutés en A (pour le pré-traitement) et en B (pour l'annotation, la classification et la prévision).

1. Quels corpus ?

Les données textuelles peuvent être sous format de SMS, de tweets, de blogs, de statuts Facebook, de commentaires dans des forum de discussion, de réponses à des questionnaires, extraits de sites web ou d'articles scientifiques collectés via des API [3] spécifiques. Il arrive aussi que des corpus différents soient associés pour mieux analyser un texte, un peu à l'image d'un lecteur qui se documente via des livres différents, de plus en plus variés, et qui apprend, grâce à cela, à reconnaître le style des écrivains. Dans notre cas, ça serait les principaux algorithmes manipulés, les diverses techniques relevées, les dernières innovations retenues, etc. Par exemple, si le choix se fait sur l'étude des traitements de semences, il suffirait de lancer une recherche web en fonction de mots clefs adéquats (traitement, fongicides, glyphosate, etc). Nous obtenons un certain nombre de documents en relation avec ces mots clés. Voir section 1.1 pour un exemple d'extraction de textes.

Ce qui est commun aux données textuelles, c'est qu'elles sont mal structurées : elles peuvent contenir des balises, des méta-data, des liens hypertextes, des mots mal orthographiés, une composition de plusieurs langues, un arrangement de chiffres, de tableaux et de figures, des émoticones, etc. Dans un premier temps, il est primordial de nettoyer le corpus : enlever les balises, transformer toutes les lettres en minuscule, traduire les mots étrangers dans le texte dans la langue étudiée, corriger les fautes d'orthographes, enlever les ponctuations, éliminer les nombres, etc. A cela, s'ajoute l'atomisation du texte. Ce n'est pas le seul pré-traitement affligé aux données. Les phrases sont réduites à leur minimum (en enlevant les mots communs appelés stop-word). C'est ce que nous montrons en section 1.2.

Une fois le corpus de textes défini, le processus d'extraction de l'information consiste en la structuration du texte. C'est la partie la plus importante : le codage du texte. Ceci consiste à représenter le texte sous forme d'un vecteur numérique, voir section 1.3.

1.1. Techniques d'extraction (web scraping)

Construire sa propre base de données passe par des applications et des fonctions spécifiques (voir par exemple les API de Twitter, de Facebook ou de Google pour récupérer des données textuelles). Il est en effet possible de collecter des données textuelles directement sur des sites spécifiques, comme les sites d'e-commerce. Nous pouvons ainsi assembler puis télécharger un catalogue de produits (machines agricoles, engrais, semences) avec des caractéristiques de chaque élément (des attributs comme la marque, la référence, la catégorie, la description ou toute autre information relative au produit).

Avec le choix d'utilisation du logiciel Python, il y a principalement les librairies Scrapy (Myers and McGuffee 2015), BeautifulSoup (Richardson 2007) ou urllib2 (Lawson 2015) qui permettent ce type de traitement. Pareillement, nous pouvons appliquer la librairie Selenium (Avasarala 2014) qui permet une extraction structurée de textes, en spécifiant le site web et le type de navigateur. Nous avons tiré un premier exemple depuis le site Scopus où nous avons lancé la requête suivante

TITLE-ABS-KEY(agriculture) AND (LIMIT-TO(PUBYEAR,2019)) AND (LIMIT-TO(ACCESSTYPE(OA)))

afin de collecter les sources des travaux sur l'agriculture, publiés en libre accès (*Open Access* ou OA) du 1er janvier au 30 avril 2019. Dans la figure 1 nous avons projeté sur une map-monde les pays d'origine de ces travaux.

Par ailleurs, il y a moyen de travailler sur le flux de données sur Twitter : ce sont des statuts d'où nous pouvons tirer l'adresse IP (et donc la région ou la localisation géographique) et l'évolution temporelle

3. API : Application Programming Interface. C'est un ensemble de classes, de fonctions et de méthodes accompagnant un logiciel ou un service web.

FIGURE 1: Le nombre de publications scientifiques sur l'agriculture, selon le pays ou le territoire. Nous avons limité la requête au premier trimestre de 2019 et aux journaux en libre accès. Nous avons trouvé 1892 documents référencés sur Scopus, avec du premier au dernier classement : les Etats Unis avec 360 publications retenues, la Chine, l'Angleterre, l'Indonésie, l'Allemagne, l'Italie, l'Espagne, l'Inde, les Pays-Bas et le Brésil avec 79 publications en libre accès sur l'agriculture

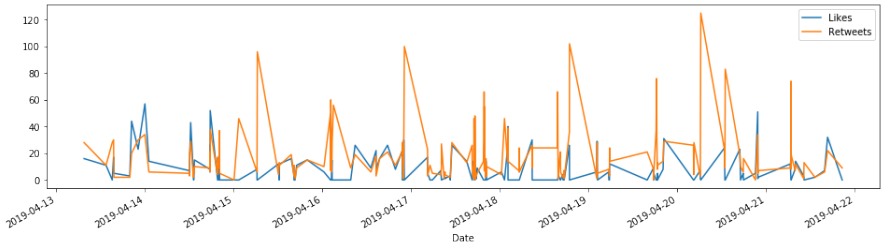

FIGURE 2: Comportement des utilisateurs de Twitter sur le mois d'avril 2019 qui s'intéressent à la communauté des militants contre les organismes génétiquement modifiés connu sous le nom *GMOfree*. Nous voyons une corrélation entre le nombre d'appréciations (Likes) et le nombre de partages (Re-tweet)

(selon la saison, le jour de la semaine, l'heure de la journée), suivant ce que nous souhaitons appliquer au corpus. Par exemple, partant de tweets collectés autour de dix régions des Etats Unis, une estimation des dates de semence pour chaque état a été finalisé, les nouveaux enjeux agricoles (e.g. les conditions météorologiques, les ravageurs des cultures) et les orientations des activités de vulgarisation et de sensibilisation de ces régions ont été tirés pour chaque culture (Zipper 2018).
Ainsi, la saison ou l'horaire des publications sur le réseau internet a une influence sur l'analyse de l'information collectée. Pour illustrer cela, nous avons choisi un exemple de tweets liés à la communauté GMOfree [4]. Les tweets sont extraits durant le premier trimestre de l'année 2019. La base analysée contient une centaine de tweets. La figure 2 donne l'évolution des appréciations et des partages des divers tweets sur le mois d'avril 2019.

Notons que les personnes qui visitent un site sont identifiées via une liste d'intérêts (basée sur les mêmes caractéristiques et enregistrant toute autre information de navigation), et ceci sert par exemple dans la détection des influenceurs (Ghosh, Viswanath, Kooti, Sharma, Korlam, Benevenuto, Ganguly, and Gummadi 2012). Ce sont aussi des informations utilisées dans la modélisation du comportement des visiteurs, comme c'est la cas dans (Ray 2017) ou dans (Ram, Vishal, Dhanalakshmi, and Vidya 2015) pour l'étude de la régulation de l'eau grâce aux capteurs intelligents en agriculture. D'autres comme (Hunt 2007) utilisent des études classiques, sur le terrain suite à des questionnaires, ou ciblent une recherche

4. Militant contre les organismes génétiquement modifiés

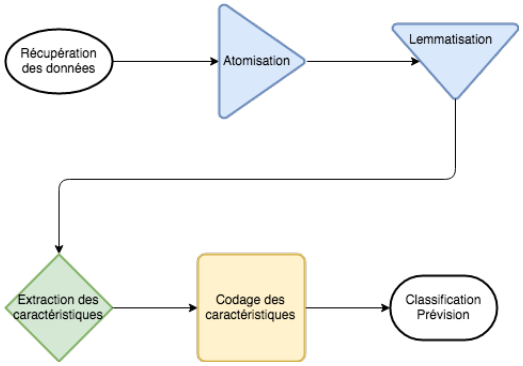

FIGURE 3: Méthodologie de fouille des données couramment utilisée dans la littérature. Après le formatage des données, le texte est transformé en une suite de phrases. C'est la segmentation. Nous procédons à la recherche de la partie d'un texte qui a le plus de signification. C'est l'extraction des caractéristiques. A cela, nous pouvons ajouter la reconnaissance de mots clés liés à la thématique étudiée. La classification peut se faire en se basant sur les groupes des verbes

spécifique de textes scientifiques sur l'agriculture (Figueroa-Rodríguez, Álvarez-Ávila, Hernández Castillo, Schwentesius Rindermann, and Figueroa-Sandoval 2019).

1.2. Pré-traitement des données

La nature des données textuelles massives, qui est liée à des objets non structurés (provenant d'une extraction de concepts, des constructions d'ontologies), fait que l'intervention des linguistes via les analyses du langage naturel et de la sémantique, est souvent nécessaire.
Pourtant, nous sommes amenés à utiliser les outils de fouille (data mining) qui fonctionnent avec des données structurées, afin de rendre l'analyse réalisable. Pour les mettre sous une forme adéquate, nous privilégions la transformation des textes pour les rendre mieux structurés. Une fois nous disposons d'un corpus, nous procédons à la mise en place d'un dictionnaire (i.e. un ensemble de mots pertinents). Pour cela, nous appliquons les opérations de traitement du langage naturel (NLP) détaillées ci-dessous : la normalisation, l'atomisation, la suppression des mots inutiles, et enfin le retour à la racine des mots. La figure 3 détaille ce processus de pré-traitement des données textuelles.

1.2.1. Normalisation du texte

La normalisation d'un document consiste à rendre l'orthographe le plus uniforme que possible. Les opérations à effectuer sont par exemple la conversion en minuscule (Bioud 2006), la transformation des chiffres en mots, la suppression de toute sorte de ponctuation et des espaces de trop entre les mots, la transformation des abréviations en groupes de mots, etc.

1.2.2. Atomisation

En anglais, nous parlons de *Tokenization*. Il s'agit de convertir une phrase en un groupe de mots (verbes, noms, nombres). Après cela, une sélection de mots s'opère. Nous enlevons les mots inutiles, qui n'apportent pas un plus au sens de la phrase (the, a, on, de, etc) ou *mots vides*. Ces mots sont appelés *stop word*. Par exemple, nous utilisons pour cela la librairie NLTK (Loper and Bird 2002; Bird et al. 2009) sur Python.

1.2.3. Racine des mots

Pour revenir à la racine du mot, il y a plusieurs algorithmes qui consistent à éliminer les terminaisons morphologiques communes des mots. Notamment, l'algorithme Porter stemming (Porter 2001) et l'algorithme Lancaster stemming (Hooper and Paice 2005). Il est aussi possible d'utiliser une autre technique se basant sur le sens du mot comme dans NLTK (WordNet Lemmatizer), spaCy, TextBlob, Pattern, gensim, Stanford CoreNLP, MBSP, GATE, Illinois Lemmatizer, et DKPro Core. Voir (Solberg 2012) pour une revue complète sur la construction d'un corpus.

Le système d'indexation des termes produit un résultat pertinent sur l'analyse de textes. Aussi, au dictionnaire de mots, s'ajoute la pondération des termes qui affecte une mesure sur l'importance d'un mot particulier dans la description d'un produit (par rapport aux descriptions des autre produits). La méthode la plus fréquemment utilisée est la pondération TF-IDF ou *Term Frequency-Inverse Document Frequency* comme expliqué dans (Salton and Buckley 1988). Par exemple, si nous reprenons la base de tweets collectée dans la section 1.1, et suite au pré-traitement du corpus, nous obtenons les mots les plus significatifs. Ces mots que nous voyons apparaître dans le nuage de la figure 4 sont *cancer*, *Roundup* (qui est la principale marque produisant du glyphosate). Nous notons la présence du mot Vietnam (qui a interdit le glyphosate en mars 2019). L'idée fondamentale de cette démarche est qu'une personne extérieure au domaine d'étude puisse se faire une idée générale de la base simplement en regardant les nuages de mots.

FIGURE 4: Nuage de mots généré d'une base de tweets, extraits sur la période allant de 2016 à 2019, suivant les comptes américains, canadiens et européens de @GMOfree. Nous ne conservons que les mots dont la fréquence d'apparition dépasse un certain seuil

1.3. Représentation des données textuelles

Il est important de regrouper le texte en phrases, les phrases en mots, les mots par thèmes (traitements, machineries, semences, nourritures). Ainsi, chaque texte est représenté par un groupe de mots et ces derniers sont symbolisés par un vecteur (avec leur fréquence d'apparition) dans le dictionnaire lié au corpus. Structurer l'information sous forme de vecteur permet de les fouiller pour relever, par exemple, les mots les plus fréquents. Dans (Agard 2007), le vecteur est construit dans le but de représenter le contenu de la phrase via un vecteur de poids qui se calcule sur l'apparition du terme dans le texte. Nous parlons de *word embedding* ou d'intégration de mots. Le principe de cette procédure est le suivant : chaque coordonné du vecteur représente un mot du texte.

Par exemple, si nous considérons les deux phrases `P1` et `P2` suivantes :
`P1 : On a ajouté de l azote.`
`P2 : On a une carence en azote.`
Alors, le vocabulaire formant ces phrases est :
`V = {on, a, ajouté, de, l, azote, une, carence, en}`.
Les deux phrases pourraient être représentées par les vecteurs suivants :
`P1 = [1,1,1,1,1,1,0,0,0]`
`P2 = [1,1,0,0,0,1,1,1,1]`.
Ce modèle simple et efficace est appelé *Bag of words* ou BoW. Il donne les mots qui sont présents et dans quelle mesure ils sont présents dans des documents codés (sans aucune information sur l'ordre).
Il existe de nombreuses façons d'étendre cette méthode simple, à la fois en clarifiant mieux ce qu'est un mot et en définissant ce qu'il faut coder pour chaque mot du vecteur. Ceci permet de décrire la présence de mots dans les données textuelles : par exemple, deux champs de texte similaires contiennent le même type de mots, et auront donc un sac de mots analogues. De plus, le texte seul nous permet d'apprendre sur la signification du document. BoW est la technique la plus populaire puisqu'à partir d'un vocabulaire,

elle permet de choisir une mesure de présence du mot dans ce paquet de mots. Nous obtenons des matrices avec en ligne la phrase, en colonne les mots et les coefficients sont des 0 et des 1 pour présent ou absent dans le dictionnaire des mots.

D'autre part, la fréquence d'un mot T, notée TF, est définie comme le rapport entre le nombre de fois que ce terme T apparaît dans la phrase et le nombre total de mots dans cette phrase. Un mot T qui apparaît un grand nombre de fois dans tous les documents, n'est pas d'une grande utilité. Par conséquent, nous utilisons l'IDF (la fréquence inverse des documents) qui est le logarithme du rapport entre le nombre total de phrases et le nombre de phrases dans lesquelles ce mot T est présent. En général, pour un terme donné, nous calculons la quantité TF-IDF qui est le produit de TF et de IDF du mot. A cela, nous ajoutons le sac de mots (BoW) qui fait référence à la représentation du texte.

Nous pouvons également utiliser des séquences de n mots adjacents extraits d'une séquence de mots. Cette combinaison est appelée n-gramme et est employée pour la catégorisation des textes (Fürnkranz 1998). Les n-grammes avec n=1 sont appelés uni-gramme. De même, les bi-grammes (n=2), les tri-grammes (n=3) et ainsi de suite peuvent également être utilisés (jusqu'à la phrase complète). Le principe de base des n-grammes est qu'ils capturent la structure du langage, comme la lettre ou le mot qui est susceptible de suivre celui donné. Plus n est élevé, plus nous donnons de l'importance au contexte de la phrase. La longueur optimale dépend de l'application recherchée : c'est un juste équilibre entre les différences importantes dans le texte et les connaissances générales de ce dernier. Voir par exemple (Socher, Perelygin, Wu, Chuang, Manning, Ng, and Potts 2013) pour capturer l'effet de la négation dans une phrase.

A ces méthodes intuitives et simples, nous pouvons ajouter des méthodes plus sophistiquées et plus variées. Il y a Word2Vec, Glove et FastText où l'incorporation du mot est l'une des représentations les plus populaires du vocabulaire. L'idée est de capturer le contexte d'un mot dans un document, la similarité sémantique et syntaxique, la relation avec d'autres mots, etc. Word2Vec est l'une des techniques les plus populaires d'apprentissage de l'intégration de mots à l'aide de réseaux neuronaux peu profonds. Il a été développé chez Google (Mikolov, Sutskever, Chen, Corrado, and Dean 2013). La sélection de caractéristiques passe par exemple par les méthodes SkipGram et CBOW (Kou, Li, and Baldwin 2015). D'autre part, GloVe (Pennington, Socher, and Manning 2014) est un algorithme d'apprentissage non supervisé permettant d'obtenir des représentations vectorielles de mots. La formation porte sur les statistiques globales de co-occurrence mot-mot issues d'un corpus, et les représentations résultantes présentent d'intéressantes sous-structures linéaires de l'espace vecteur-mot. Enfin, FastText (Mikolov, Grave, Bojanowski, Puhrsch, and Joulin 2018) est une bibliothèque légère, gratuite et à code source ouvert, qui permet aux utilisateurs d'apprendre les représentations et les classificateurs de texte. Cela fonctionne sur du matériel générique standard. Les modèles peuvent ensuite être réduits en taille (Joulin, Grave, Bojanowski, and Mikolov 2016; Bojanowski, Grave, Joulin, and Mikolov 2017).

Comme mentionné ci-dessus, word2vec utilise une architecture de type réseau de neurones. mais certains codages utilisent des outils de réseaux de neurones profonds comme dans (Bianchini and Scarselli 2014). La recherche dans ce domaine est effectivement très riche : voir (Howard and Ruder 2018) qui présente un parmi les derniers algorithmes utilisés dans la fouille de données textuelles. Voir aussi le codage contextuel dans (Farhan, Wang, Huang, Wang, Wang, and Jiang 2016) utilisant la technique de fenêtre dynamique sur les mots. Le plus important dans ce codage étant de garder le contexte du mot (pour ne pas perdre la signification dans la phrase).

2. Analyse sentimentale

Le métier d'agriculteur est très complexe. Il englobe la gestion quotidienne de l'exploitation (choix des cultures, rendement des parcelles, contraintes en machinerie et en ressources humaines) et une haute technicité pour un suivi de l'évolution des cultures, la détection des maladies à temps, l'utilisation raisonnée des ressources, etc. Les exploitants agricoles doivent être informés sur les variétés de semences

qui s'adaptent le mieux à leurs sols. Ils doivent connaître les effets des traitements ainsi que les résistances.

C'est dans ce contexte qu'intervient l'analyse sentimentale et plus particulièrement l'analyse de polarité (négative, neutre, positive) et l'analyse d'intensité (faible, moyenne, forte). Sur un corpus donné, ceci permet de construire des alertes via le nombre absolu et relatif de mots négatifs qui paraissent dans un message, un niveau de critique : mot neutre, mot alarmant, mot extrêmement alarmant, une indication de la présence ou de l'absence de ces alertes, etc. Par exemple le degré d'urgence d'un message se calcule en additionnant le nombre de mots critiques, multipliées par leurs niveaux de critique respectifs. Plus le résultat est élevé, plus l'énoncé est alarmant. Dans la section 2.1 nous donnons un résultat d'analyse sentimentale du corpus tiré d'un site de e-commerce.

L'objectif est de comprendre le comportement des agriculteurs ou des utilisateurs de sites de e-commerce, de leur proposer du contenu adapté à leurs attentes, de prédire leurs besoins futurs (selon les semences achetées, la saison, la localisation géographique de la parcelle exploitée). Il y a des avantages et des inconvénients dans les systèmes basés sur du contenu : suivi des utilisateurs actifs, création d'un profil d'utilisateurs, prise en compte rapidement d'un nouveau produit mis sur le marché. Les inconvénients, pour cette approche basée sur le contenu, est la nécessité d'une quantité importante d'informations, d'avoir des utilisateurs cloisonnés dans des profils spécifiques, etc. La section 2.2 présente les algorithmes qui seront utilisés en apprentissage automatique. La section 2.3 donne les performances de ces algorithmes sur notre corpus d'étude.

Aussi, il s'agira d'extraire le corpus, et d'annoter les descriptions. Puis, nous entamons la classification des documents sur la base d'entraînement. Ensuite, la démarche à suivre consiste à effectuer des prévisions pour les nouveaux documents, afin de classer correctement les descriptions en trois types (positif, négatif ou bien neutre). La méthodologie simplifiée est présentée dans la figure 5 ; la base de test étant celle qui permettra de mesurer les performances des divers classificateurs.

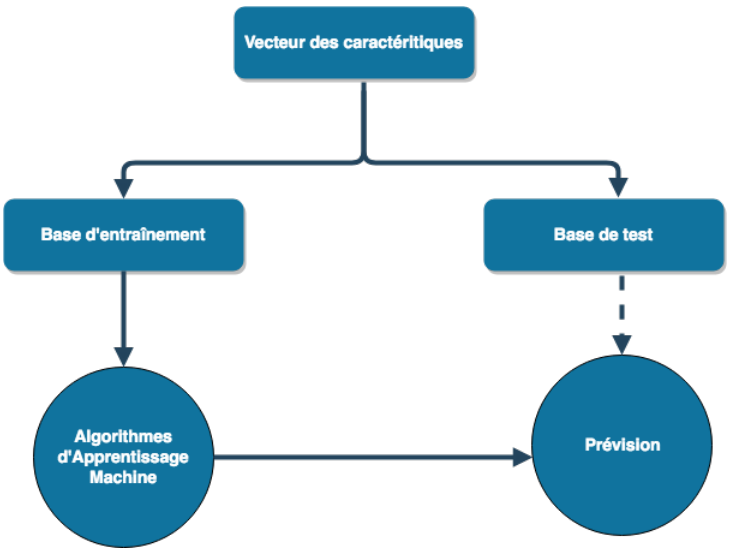

Figure 5: Modèle simplifié pour l'apprentissage machine

2.1. Exemple d'un site de e-commerce

Pour illustrer les techniques présentées dans la section précédente, nous avons choisi de travailler sur un corpus partant du site web agriconomie.com.

Nom	Description
Urée 46 Granulé - Granulé	L'Urée 46 est l'engrais azoté idéal pour toutes les cultures exigeantes en azote. Elle contient 46 unités d'azote uréique soluble dans l'eau, avec une forme de granulé. L'azote est un élément essentiel au bon développement des plantes: il est présent dans les protéines, les acides nucléiques et la chlorophylle. Il s'agit donc...
Sulfate d'Ammoniaque ...	De part ces caractéristiques techniques (diamètre, densité), son diamètre moyen élevé et sa haute densité lui permettent d'être épandu en grande largeur. Observations notables: Pour les céréales : le nombre de talle est réduit Pour le colza : On peut observer une déficience en...
⋮	⋮

Table 1: Un aperçu de la base de données récupérée depuis le site web http ://agriconomie.com. Il y a deux colonnes, la première représente le nom du produit (traitement, engrais, semence) et la seconde colonne donne la description du produit tel que trouvé sur le site web

Nous avons tiré un exemple d'une base de 506 produits. Chaque produit est donné par son nom ainsi que sa description comme indiqué dans le tableau 1. Un traitement préliminaire du texte est indispensable : normalisation, atomisation, suppression de stop-word de la langue française et recherche des racines des mots.

Grâce au pré-traitement de la base, nous pouvons avoir des informations sur la longueur moyenne des descriptions, les mots les plus fréquents, la présence d'un mot dans les descriptions, etc. Les deux nuages de mots de la colonne nom et de la colonne description sont donnés dans la figure 6. Les mots les plus fréquents dans chaque nuage de mots (en dehors des stop word) illustre les produits vendus sur ce site de e-commerce ainsi que leurs descriptions.

(a) (b)

Figure 6: La figure 6a représente le nuage de mots pour les noms des produits. Les mots qui reviennent le plus souvent dans le nom des produits et le type d'engrais sont liquide, granulé ou en bouchon. La figure 6b donne le nuage de mots pour la description des produits. Des conseils émanent des descriptions comme le stade d'administration du traitement ou les éléments nutritifs constituant les produits

En étudiant automatiquement le corpus, nous pouvons attribuer une classe à chaque description : selon que les mots utilisés soient neutres, positifs ou négatifs. Nous obtenons ainsi trois classes comme le montre la figure 7. Une description neutre du produit, une description positive et une description négative. En effet, l'utilisation des produits phytosanitaires ou des insecticides nécessite souvent une attention particulière. L'étude des descriptions données dans le corpus permet de détecter de manière systématique les produits dangereux (classés dans négatif) ou ayant des descriptions neutres ou positives (absence de termes alarmant dans la description du produit). Après cela, et selon la classification de la description, une alerte pourrait être générée selon la gravité du produit et ses effets sur le sol, la plante ou la personne qui le manipulerait. Mais pour réussir cela, encore faut-il avoir des documents étiquetés (c'est là où intervient l'analyse sentimentale).

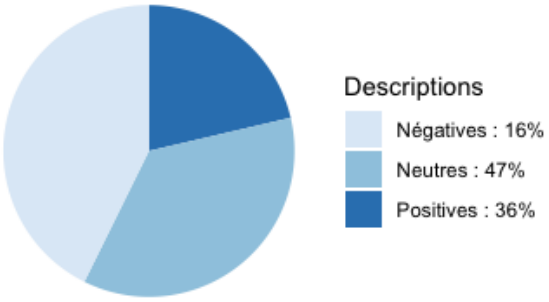

FIGURE 7: Annotation des descriptions suite à une analyse sentimentale (ou de polarité). Nous trouvons qu'une majorité des descriptions a une tonalité neutre, ce qui est appréciable lorsqu'il s'agit de vente d'insecticides ou d'herbicides. Les descriptions sont à 16% négatives, ce qui indiquerait une alerte. Le reste des descriptions est positif (36%)

2.2. Classification de textes

L'analyse d'opinion comme présentée dans (Nimirthi, Venkata Krishna, Obaidat, and Saritha 2019) sur des tweets collectés (ou des SMS) passe par une classification des documents. Il y a trois types d'approches : supervisées, non supervisées ou hybrides. En effet, après les mesures à appliquer pour le traitement des données (atomisation, suppression des stop-word et recherche des racines des mots) et une fois que les mots sont plongés dans des vecteurs, nous pouvons extraire les caractéristiques, qui sont l'entrée des outils de classification des genres (Balikas, Moura, and Amini 2017).

Nous présentons quatre exemples d'algorithmes d'apprentissage : SVM, les réseaux de neurones artificiels, les forêts aléatoires et l'algorithme Naive Bayes. Ce sont des méthodes largement utilisées en apprentissage machine (Patel and Patel 2014; Ramesh, Ramar, et al. 2011) qui s'adaptent bien au cas de la fouille textuelle.

2.2.1. Machines à vecteurs de support (SVM)

Les machines à vecteurs de support (connu sous le nom SVM) sont des classificateurs linéaires généralisés. Ils minimisent simultanément l'erreur de classification et maximisent la marge géométrique (appelés classificateurs de marge maximale). L'algorithme SVM a été introduit pour la première fois par (Vapnik 1963), puis étendu pour inclure l'astuce du noyau (Vapnik 1999). La motivation d'un SVM est de trouver un hyperplan dans un espace de grande dimension pouvant être utilisé pour la classification. Le but est de trouver une bonne séparation par l'hyperplan en gardant la plus grande distance (marge fonctionnelle). L'intuition est qu'une marge fonctionnelle plus grande entraîne une erreur de généralisation plus faible pour le classificateur. Si x est le vecteur de données et y son label, w le vecteur normal à l'hyperplan $wx - b = 0$, alors il s'agira de résoudre le problème d'optimisation (vu en terme de lagrangien) :

$$arg\ min_{w,b}\ max_{\alpha \geq 0}\ \left\{ \frac{1}{2}||w||^2 - \sum \alpha_i (y_i (w.x_i - b)) \right\}$$

2.2.2. Réseaux de neurones (NN)

Les réseaux de neurones sont utilisés dans les diverses étapes de fouille de texte. Par exemple, nous avons vu dans la section 1.3 que le modèle prédicatif efficace (i.e. qui tient compte du contexte) pour l'apprentissage des *word embeddings* à partir de texte brut est donné par Word2Vec. Cette méthode utilise un réseau de neurones à deux couches cachées. Un réseau de neurones contient au moins deux couches : une couche d'entrée x et une couche de sortie y. Il peut contenir des couches intermédiaires de neurones qu'on appelle couches cachées (Hassoun et al. 1995).

Les neurones dans les diverses couches sont reliés par des arcs. Chaque arc a un poids appelé *poids synaptique*. L'objectif de l'algorithme est de chercher les poids synaptiques optimaux qui assurent que la sortie du réseau de neurones $y = f(\sum w_{i,j}\ x_j)$ corresponde à ce qui est observé. La fonction f est appelée *fonction d'activation*. Dans notre cas, nous avons choisi la fonction ReLu définie par $f(z) = 0$ si $z < 0$ et $f(z) = z$ sinon. Mais il y a une multitude de fonctions d'activation comme la fonction sigmoïde ou la fonction tanh.

Plus généralement, il y a moyen d'utiliser des réseaux de neurones graphiques dans le text mining, surtout lorsque les données sont volumineuses (Scarselli, Gori, Tsoi, Hagenbuchner, and Monfardini 2009). Mais, les réseaux de neurones sont des boîtes noires qui donnent des modèles avec un risque de sur-apprentissage.

2.2.3. Forêts aléatoires (RF)

L'algorithme d'apprentissage via les forêts aléatoires (connu sous l'abréviation RF pour *Random Forest*) est une combinaison de plus d'un classificateur, avec de nombreux avantages tels que l'efficacité, plus de variables d'entrée manipulées, l'importance des variables, la robustesse au bruit ainsi que les valeurs aberrantes (Liaw, Wiener, et al. 2002). Il aide à classer les variables en régression ou en classification. Dans (Lebourgeois, Dupuy, Vintrou, Ameline, Butler, and Bégué 2017) il est utilisé pour la classification de données satellites des cartes d'utilisation des terres d'une zone agricole de petits exploitants à Madagascar. Dans (Achat, Pousse, Nicolas, Brédoire, and Augusto 2016) c'est une compilation de la littérature qui est utilisée dans la compréhension du cycle du Phosphate dans les sols de forêts françaises.

2.2.4. Algorithme Naive Bayes (NB)

Cet algorithme se base sur le théorème de Bayes qui est fondamentalement lié aux probabilités conditionnelles. Si A et B sont deux évènement, alors la probabilité que l'évènement A se réalise sachant que l'évènement B s'est déjà réalisé est donnée par

$$P(A|B) = \frac{P(B|A)\ P(A)}{P(B)}\ .$$

Lorsqu'il y a plusieurs variables (comme c'est le cas dans un texte), l'approche de l'algorithme Naive Bayes consiste à prendre en considération ces variables indépendamment les unes des autres. L'algorithme s'appuie sur la fréquence des occurrences des mots dans le texte pour en définir la catégorie (Ting, Ip, and Tsang 2011). Lors de la classification, l'algorithme suppose que les mots du texte apparaissent indépendamment les uns des autres. Il s'agit d'une hypothèse forte et le terme *Naive* vient de cette supposition.

2.3. Mesures d'évaluation

Nous décrivons ci-dessous un ensemble de mesures d'évaluation standard largement utilisées dans l'analyse des données. Ces mesures servent à quantifier les performances d'un algorithme de classification binaire supervisé, quelle que soit la tâche. Les mêmes mesures peuvent toujours être utilisées pour la classification multi-classe en divisant l'évaluation en C classifications binaires, où C est l'ensemble des classes. Ici $C = 3$ pour la description négative, neutre ou positive. La métrique de performance la plus élémentaire est l'exactitude *Accuracy*, qui mesure simplement la proportion d'instances prédites correctement par le classificateur :

$$Accuracy\ =\ \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{FP} + \mathrm{TN} + \mathrm{FN}}$$

où TP sont les vrais positifs, FP sont les faux positifs, TN sont les vrais négatifs et FN sont les faux négatifs.

La précision P mesure la proportion d'instances classées positivement correctement prédites par le classificateur, tandis que le rappel R mesure la proportion d'instances positives dans les données que le classificateur peut prédire. Ils sont définis comme suit :

$$P = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \quad ; \quad R = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \ .$$

Il est souvent pratique de combiner P et R en utilisant la moyenne harmonique ; cette mesure est connue sous le nom de $F_1 score$:

$$F_1 score = \frac{2PR}{P + R} \ .$$

Nous évaluons notre méthodologie de classification intrinsèquement en utilisant des mesures de performance standard (précision, rappel, exactitude et $F_1 score$).

Algorithme	Accuracy	P	R	$F_1 score$
RF	0.85	0.87	0.85	0.86
SVM	0.85	0.86	0.85	0.85
NB	0.78	0.82	0.78	0.79
NN	**0.87**	0.87	0.87	0.87

Table 2: Performance des algorithmes de classification. Nous avons testé plusieurs architectures pour le perceptron multicouches. Le plus performant est le réseau de neurones avec 2 couches cachées, contenant 5 neurones pour la première couche et 10 neurones pour la couche suivante

Appliquée à la base de données récupérée depuis le site agriconomie.com (annotée grâce à l'analyse sentimentale), nous comparons les quatre algorithmes d'apprentissage automatique : SVM, NB, NN et RF. Nous obtenons des performances détaillées dans le tableau 2, qui indique l'exactitude, la précision, le rappel et le $F_1 score$. Nous remarquons que c'est l'algorithme basé sur les réseaux de neurones qui est le plus performant, comparé aux autres, et compte-tenu des données utilisées.
Ces résultats pourraient être améliorés si la base de données était plus grande. Les performances de l'algorithme Naive Bayes pourraient être perfectionnées en suivant le procédé détaillé dans (Kim, Han, Rim, and Myaeng 2006). De plus, il est important de relever qu'à ce stade, le principal soucis dans l'annotation automatique de texte, utilisant l'analyse sentimentale, c'est l'ambiguïté de certaines phrases ou de certains mots (que nous retrouvons dans la classe neutre). Si la manière d'annoter le corpus était plus fine, la classe neutre serait sans doute réduite au profit des classes positif et négatif. L'optimisation de cette partie est donné par exemple dans (Weichselbraun, Gindl, and Scharl 2013).

Pour résumer, notons que les méthodes utilisées relèvent de l'apprentissage machine. D'autres utilisent des architectures de types réseaux de neurones profonds. Ce sont des techniques développées pour l'insertion contextuelle utilisée dans la prédiction (Li, Jing, Tong, Yang, He, and Chen 2017). Il y a aussi (Gardner, Grus, Neumann, Tafjord, Dasigi, Liu, Peters, Schmitz, and Zettlemoyer 2018) qui utilise de l'apprentissage profond, avec des architectures se basant sur des réseaux de neurones comme le bi-LSTM (Baziotis, Pelekis, and Doulkeridis 2017). Une librairie disponible sur git-hub est dans (Pressel, Choudhury, Lester, Zhao, and Barta 2018).

3. Conclusion

Le monde agricole est confronté à des problèmes tels que le manque de successeurs et l'augmentation du nombre d'agriculteurs âgés. Dans (Shimoguchi, Inaizumi, Yasue, and Omuro 2015), c'est une solution qui est apportée pour attirer et offrir des possibilités d'apprentissage aux jeunes générations pour qu'elles deviennent de nouveaux agriculteurs. Ce que nous recherchons ce sont des modèles d'optimisation de la gestion agricole.

Elaborer puis appuyer la prise de décisions concernant les activités agricoles, notamment la plantation, la récolte, la manutention, le séchage et l'entreposage devient indispensable, surtout que les nouveaux exploitants ne sont pas forcément familiers avec des techniques agricoles basées sur l'expérience. Les travaux décrits dans ce travail démontrent clairement l'intérêt et l'apport de l'intelligence artificielle et la fouille textuelle au domaine de l'agriculture numérique ou dans les sciences agronomiques.

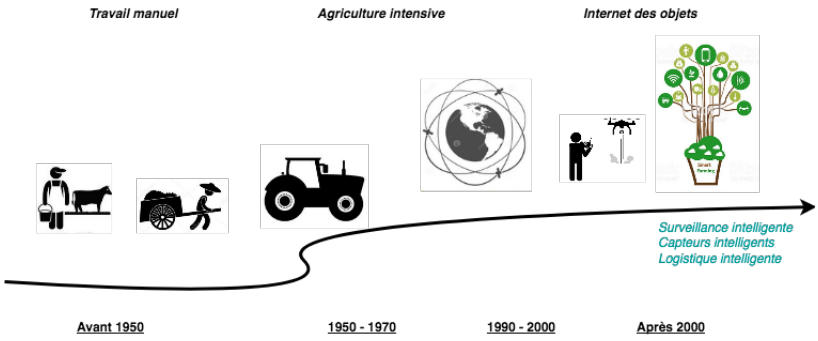

FIGURE 8: Evolution des diverses techniques dans le domaine agricole

Les études les plus récentes portent également sur l'Internet des objets parce qu'un nombre de plus en plus important de capteurs et d'objets intelligents sont reliés à l'activité agricole. La figure 8 montre l'évolution des techniques agricoles, et l'utilisation des outils de calculs modernes influent sur le rendement de l'agriculture d'aujourd'hui. Dans (Liao, Rodriguez, Diesner, and Lin 2015), l'exploration de texte est utilisée pour identifier les calendriers de plantation et de récolte pour différentes cultures dans différentes régions des USA. Ainsi, les récoltes peuvent être optimisées lorsqu'on met en place un moyen d'extraction de l'information en temps quasi réel afin de déterminer le moment de l'ensemencement des cultures et les calendriers de récoltes à partir de bases de données accessibles au public. Aussi, nous avons vu que les alertes des maladies et le calendrier des traitements peut être obtenu systématiquement grâce à une fouille de bases de données judicieuses.

D'autre part, le travail préliminaire que nous avons mené sur la fouille de sites web pourrait être utilisé afin de développer un moteur d'avis et de conseils pour une plate-forme de vente agricole. Des recommandations basées sur le contenu, sur la collaboration (notes des utilisateurs) et hybrides (Sundermann, Domingues, Sinoara, Marcacini, and Rezende 2019) est en cours de réalisation.

Remerciements. Nous tenons à remercier Asma Bouhafs pour ses remarques judicieuses et Naceur Tej pour son regard avisé de professionnel en agriculture, curieux des nouvelles technologies. Aussi, ce travail s'est fait en marge des encadrements réalisés au sein de l'école supérieure de la statistique et de l'analyse de l'information de Tunisie sur la fouille de textes. Nous saluons particulièrement Sayf Chagtmi, Samar Hannachi et Ahmd Saad. Par leur curiosité et leurs diverses questions, nos élèves-ingénieurs nous ont amené à nous intéresser à cet aspect de l'intelligence artificielle et de l'apprentissage machine.

Achat, D. L., Pousse, N., Nicolas, M., Brédoire, F., Augusto, L., 2016. Soil properties controlling inorganic phosphorus availability : general results from a national forest network and a global compilation of the literature. Biogeochemistry 127 (2-3), 255–272.

Agard, B., 2007. Extraction de connaissances à partir de données textuelles - vue d'ensemble. In : 10ème Colloque National AIP PRIMECA, La Plagne. pp. 1–10.

Avasarala, S., 2014. Selenium WebDriver practical guide. Packt Publishing Ltd.

Balikas, G., Moura, S., Amini, M.-R., 2017. Multitask learning for fine-grained twitter sentiment analysis. In : Proceedings of the 40th international ACM SIGIR conference on research and development in information retrieval. ACM, pp. 1005–1008.

Basnet, B., Bang, J., 2018. The state-of-the-art of knowledge-intensive agriculture : A review on applied sensing systems and data analytics. Journal of Sensors 2018.

Baziotis, C., Pelekis, N., Doulkeridis, C., 2017. Datastories at semeval-2017 task 4 : Deep lstm with attention for message-level and topic-based sentiment analysis. In : Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017). pp. 747–754.

Bianchini, M., Scarselli, F., 2014. On the complexity of neural network classifiers : A comparison between shallow and deep architectures. IEEE transactions on neural networks and learning systems 25 (8), 1553–1565.

Bioud, M., 2006. Une normalisation de l'emploi de la majuscule et sa représentation formelle pour un système de vérification automatique des majuscules dans un texte. Ph.D. thesis, Sciences du langage Besançon, thèse de doctorat dirigée par Cardey-Greenfield, Sylviane.

Bird, S., Klein, E., Loper, E., 2009. Natural language processing with Python : analyzing text with the natural language toolkit. " O'Reilly Media, Inc.".

Bojanowski, P., Grave, E., Joulin, A., Mikolov, T., 2017. Enriching word vectors with subword information. Transactions of the Association for Computational Linguistics 5, 135–146.

Burnham, J. F., 2006. Scopus database : a review. Biomedical digital libraries 3 (1), 1.

Chebrolu, N., Lottes, P., Schaefer, A., Winterhalter, W., Burgard, W., Stachniss, C., 2017. Agricultural robot dataset for plant classification, localization and mapping on sugar beet fields. The International Journal of Robotics Research.

Comer, C., 2016. Don't ruffle feathers. timid changes f in the representation of women in the agricultural press in brittany [ne pas faire mauvais genre .les timides mutations de la représentation des femmes dans la presse agricole bretonne]. Etudes Rurales 198 (2), 77–95.

Da Costa, E., Tjandrasa, H., Djanali, S., 2018. Text mining for pest and disease identification on rice farming with interactive text messaging. International Journal of Electrical and Computer Engineering 8 (3), 1671–1683.

Devyatkin, D., Nechaeva, E., Suvorov, R., Tikhomirov, I., 2018. Mapping the research landscape of agricultural sciences. Foresight and STI Governance 12 (1), 69–78.

Farhan, W., Wang, Z., Huang, Y., Wang, S., Wang, F., Jiang, X., 2016. A predictive model for medical events based on contextual embedding of temporal sequences. JMIR medical informatics 4 (4).

Figueroa-Rodríguez, K. A., Álvarez-Ávila, M. d. C., Hernández Castillo, F., Schwentesius Rindermann, R., Figueroa-Sandoval, B., 2019. Farmers' market actors, dynamics, and attributes : A bibliometric study. Sustainability 11 (3), 745.

Fürnkranz, J., 1998. A study using n-gram features for text categorization. Austrian Research Institute for Artifical Intelligence 3 (1998), 1–10.

Gardner, M., Grus, J., Neumann, M., Tafjord, O., Dasigi, P., Liu, N., Peters, M., Schmitz, M., Zettlemoyer, L., 2018. Allennlp : A deep semantic natural language processing platform. arXiv preprint arXiv :1803.07640.

Garner, S. R., et al., 1995. Weka : The waikato environment for knowledge analysis. In : Proceedings of the New Zealand computer science research students conference. pp. 57–64.

Géron, A., 2019. Hands-on machine learning with Scikit-Learn and TensorFlow : concepts, tools, and techniques to build intelligent systems. " O'Reilly Media, Inc.".

Ghosh, S., Viswanath, B., Kooti, F., Sharma, N. K., Korlam, G., Benevenuto, F., Ganguly, N., Gummadi, K. P., 2012. Understanding and combating link farming in the twitter social network. In : Proceedings of the 21st international conference on World Wide Web. ACM, pp. 61–70.

Hassoun, M. H., et al., 1995. Fundamentals of artificial neural networks. MIT press.

Hooper, R., Paice, C., 2005. The lancaster stemming algorithm. University of Lancester.

Howard, J., Ruder, S., 2018. Universal language model fine-tuning for text classification.

Hunt, A. R., 2007. Consumer interactions and influences on farmers' market vendors. Renewable agriculture and food systems 22 (1), 54–66.

Joulin, A., Grave, E., Bojanowski, P., Mikolov, T., 2016. Bag of tricks for efficient text classification. arXiv preprint arXiv :1607.01759.

Kim, S.-B., Han, K.-S., Rim, H.-C., Myaeng, S. H., Nov 2006. Some effective techniques for naive bayes text classification. IEEE Transactions on Knowledge and Data Engineering 18 (11), 1457–1466.

Kou, W., Li, F., Baldwin, T., 2015. Automatic labelling of topic models using word vectors and letter trigram vectors. In : Zuccon, G., Geva, S., Joho, H., Scholer, F., Sun, A., Zhang, P. (Eds.), Information Retrieval Technology. Springer International Publishing, Cham, pp. 253–264.

Lawson, R., 2015. Web scraping with Python. Packt Publishing Ltd.

Lebourgeois, V., Dupuy, S., Vintrou, É., Ameline, M., Butler, S., Bégué, A., 2017. A combined random forest and obia classification scheme for mapping smallholder agriculture at different nomenclature levels using multisource data (simulated sentinel-2 time series, vhrs and dem). Remote Sensing 9 (3), 259.

Li, L., Jing, H., Tong, H., Yang, J., He, Q., Chen, B.-C., 2017. Nemo : Next career move prediction with contextual embedding. In : Proceedings of the 26th International Conference on World Wide Web Companion. WWW '17 Companion. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Switzerland, pp. 505–513.
URL https://doi.org/10.1145/3041021.3054200

Liao, W.-T., Rodriguez, L., Diesner, J., Lin, T., 2015. Improving farm management optimization : Application of text data analysis and semantic networks. In : American Society of Agricultural and Biological Engineers Annual International Meeting 2015. Vol. 4. pp. 3040–3049.

Liaw, A., Wiener, M., et al., 2002. Classification and regression by randomforest. R news 2 (3), 18–22.

Liu, Z., Chen, Y., Dai, Y., Guo, C., Zhang, Z., Chen, X., 2018. Syntactic and semantic features based relation extraction in agriculture domain. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 11242 LNCS, 252–258.

Loper, E., Bird, S., 2002. Nltk : the natural language toolkit. arXiv preprint cs/0205028.

Mikolov, T., Grave, E., Bojanowski, P., Puhrsch, C., Joulin, A., 2018. Advances in pre-training distributed word representations. In : Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018). pp. 1–4.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., Dean, J., 2013. Distributed representations of words and phrases and their compositionality. In : Advances in neural information processing systems. pp. 3111–3119.

Myers, D., McGuffee, J. W., 2015. Choosing scrapy. Journal of Computing Sciences in Colleges 31 (1), 83–89.

Măruşter, L., Faber, N., Jorna, R., Van Haren, R., 2008. A process mining approach to analyse user behaviour. In : WEBIST 2008 - 4th International Conference on Web Information Systems and Technologies, Proceedings. Vol. 2. pp. 208–214.

Nimirthi, P., Venkata Krishna, P., Obaidat, M., Saritha, V., 2019. A framework for sentiment analysis based recommender system for agriculture using deep learning approach. SpringerBriefs in Applied Sciences and Technology, 59–66.

Patel, H., Patel, D., 2014. A brief survey of data mining techniques applied to agricultural data. International Journal of Computer Applications 95 (9).

Pennington, J., Socher, R., Manning, C., 2014. Glove : Global vectors for word representation. In : Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). pp. 1532–1543.

Pivoto, D., Waquil, P. D., Talamini, E., Finocchio, C. P. S., Corte, V. F. D., de Vargas Mores, G., 2018. Scientific development of smart farming technologies and their application in brazil. Information Processing in Agriculture 5 (1), 21 – 32.

Porter, M. F., 2001. Snowball : A language for stemming algorithms.

Pressel, D. M., Choudhury, S. R., Lester, B., Zhao, Y., Barta, M., 2018. Baseline : A library for rapid modeling, experimentation and development of deep learning algorithms targeting nlp. In : Association for Computational Lingustics. pp. 1–7.

R Core Team, 2017. R : A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria.
URL https://www.R-project.org/

Ram, V. V. H., Vishal, H., Dhanalakshmi, S., Vidya, P. M., July 2015. Regulation of water in agriculture field using internet of things. In : 2015 IEEE Technological Innovation in ICT for Agriculture and Rural Development (TIAR). pp. 112–115.

Ramesh, V., Ramar, K., et al., 2011. Classification of agricultural land soils : a data mining approach. Agricultural Journal 6 (3), 82–86.

Ray, P. P., 2017. Internet of things for smart agriculture : Technologies, practices and future direction. Journal of Ambient Intelligence and Smart Environments 9 (4), 395–420.

Richardson, L., 2007. Beautiful soup documentation. April.

Ricroch, A., Harwood, W., Svobodová, Z., Sági, L., Hundleby, P., Badea, E., Rosca, I., Cruz, G., Salema Fevereiro, M., Marfà Riera, V., Jansson, S., Morandini, P., Bojinov, B., Cetiner, S., Custers, R., Schrader, U., Jacobsen, H.-J., Martin-Laffon, J., Boisron, A., Kuntz, M., 2016. Challenges facing european agriculture and possible biotechnological solutions. Critical Reviews in Biotechnology 36 (5), 875–883.

Rossum, G., 1995. Python reference manual. Tech. rep., Amsterdam, The Netherlands, The Netherlands.

Salton, G., Buckley, C., 1988. Term-weighting approaches in automatic text retrieval. Information Processing and Management 24 (5), 513 – 523.

Sanner, M. F., et al., 1999. Python : a programming language for software integration and development. J Mol Graph Model 17 (1), 57–61.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., Monfardini, G., 2009. The graph neural network model. IEEE Transactions on Neural Networks 20 (1), 61–80.

Shimoguchi, N., Inaizumi, H., Yasue, H., Omuro, K., 2015. Impact of farm-based learning practices on young farmers : Case from an organic farm in ogawa town, saitama prefecture, japan. Journal of the International Society for Southeast Asian Agricultural Sciences 21 (2), 143–167.

Silge, J., Robinson, D., 2017. Text mining with R : A tidy approach. " O'Reilly Media, Inc.".

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A., Potts, C., 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In : Proceedings of the 2013 conference on empirical methods in natural language processing. pp. 1631–1642.

Solberg, L. J., 2012. A corpus builder for wikipedia. Master's thesis, Department of Informatics, University of Oslo.

Sundermann, C., Domingues, M., Sinoara, R., Marcacini, R., Rezende, S., 2019. Using opinion mining in context-aware recommender systems : A systematic review. Information (Switzerland) 10 (2).

Ting, S., Ip, W., Tsang, A. H., 2011. Is naive bayes a good classifier for document classification. International Journal of Software Engineering and Its Applications 5 (3), 37–46.

Vapnik, V., 1963. Pattern recognition using generalized portrait method. Automation and remote control 24, 774–780.

Vapnik, V. N., 1999. An overview of statistical learning theory. IEEE transactions on neural networks 10 (5), 988–999.

Weichselbraun, A., Gindl, S., Scharl, A., 2013. Extracting and grounding contextualized sentiment lexicons. IEEE Intelligent Systems 28 (2), 39–46.

Zipper, S. C., 2018. Agricultural research using social media data. Agronomy Journal 110 (1), 349–358.

Zschocke, T., 2019. Enriching metadata in an international agroforestry research repository with funding-related information by mining grey literature. In : Procedia Computer Science. Vol. 146. pp. 112–122.

Annexes

A. Pré-traitement des données

Afin de produire sa propre base de données selon un thème d'études choisi (traitement, semence, qualité du sol, irrigation, etc), la première partie consiste à récupérer les données du web.

A.1. Extraction des données

Les librairies que nous avons utilisé pour récupérer les données textuelles du web sont importées grâce au code ci-dessous :

```
from selenium import webdriver
import pandas as pd
from selenium.common.exceptions import NoSuchElementException
import re
```

Nous avons choisi de récupérer les données depuis le site web agriconomie.com et notre navigateur web est Chrome. Le principe est de récupérer les données textes des divers pages web, de manière structurée. Les données sont enregistrées dans `data` qui a deux colonnes `nom` et `description`.

```
lien = ['https://www.agriconomie.com/engrais/',
        'https://www.agriconomie.com/semences/'
        'https://www.agriconomie.com/phyto'
        'https://www.agriconomie.com/nutrition'
        'https://www.agriconomie.com/bio'

        ]
```

En vue de manipuler les données, le mieux est d'utiliser la librairie `pandas`. La base de données comprend deux colonnes : une colonne `nom` présentant le produit (engrais, semences, etc) et une colonne `description` décrivant le produit agricole.

```
import pandas as pd
data = pd.DataFrame.from_records(data,columns = ['nom','description'])
```

A.2. Pré-traitement de la base

La préparation des données commence par la normalisation du texte. Par exemple, pour enlever les accents, nous pouvons utiliser le code suivant :

```
import numpy as np
from text_unidecode import unidecode
import unicodedata

def enlever_accents(text):
    try:
        text = unicode(text, 'utf-8')
    except (TypeError, NameError):
    text = unicodedata.normalize('NFD', text)
    text = text.encode('ascii', 'ignore')
    text = text.decode("utf-8")
    return str(text)
```

Ensuite, nous enlevons les mots qui n'ont pas d'impact sur le sens des phrases pour la langue française (stop words) ainsi que les ponctuations.

```
from nltk.corpus import stopwords
stop = stopwords.words('french')
data['stopwords'] = data['nom'].apply(lambda x:
                            len([x for x in x.split() if x in stop]))

data['nom'] = data['nom'].str.replace('[^\w\s]','')
```

Pour dessiner le nuage des mots, nous utilisons les libraires et les instructions suivantes :

```
import matplotlib.pyplot as plt
from wordcloud import WordCloud, STOPWORDS

t = ",".join(str(i) if not isinstance(i, unicode)
                        else i for i in data["description"])
stop = set(stopwords.words('french'))
wordcloud = WordCloud(background_color="white", max_words=2000,
stopwords=stop, contour_width=3, contour_color='steelblue').generate(t)

plt.imshow(wordcloud)
plt.axis('off')
plt.show()
```

B. Traitement des données

Une fois la base nettoyée, le texte converti en vecteurs, nous entamons le traitement et l'analyse des données. Trois étapes fondamentales sont suivies : l'analyse sentimentale permettant l'annotation de textes, la classification et enfin la prévision. Notons que tous les codes que nous détaillons ici pourraient être ré-écris avec les fonctions développées dernièrement sur Keras (Géron 2019). Voir par exemple le site RealPython [5] pour des exemples de classification.

B.1. Annotation

Le code suivant produit, pour chaque description, un chiffre égal à 1 si la description est positive, -1 si elle est négative et 0 si elle est neutre.

```
from textblob import TextBlob
from textblob_fr import PatternTagger, PatternAnalyzer

def clean(text):
    return ' '.join(re.sub("(@[A-Za-z0-9]+)|
     ([^0-9A-Za-z \t])|(\w+:\/\/\S+)", " ", text).split())

def sentiment(text):
    analysis = TextBlob(clean(text))
    if analysis.sentiment.polarity > 0:
        return 1
    elif analysis.sentiment.polarity == 0:
        return 0
    else:
        return -1
```

Les valeurs calculées par `sentiment` sur `data["description"]` sont enregistrées dans un vecteur `y`.

B.2. Classification

Les codes de classification de textes sont disponibles sur le net [6]. L'avantage de coder en Python est d'avoir une communauté très active sur le web et des codes en libre accès qu'il suffirait d'adapter à nos données. Ici, `X` représente la colonne de description et `y` la classe associée (0,1 ou -1) selon que la description du produit est respectivement (neutre, positive, négative).

```
from sklearn.feature_extraction.text import TfidfVectorizer
tfidfconverter = TfidfVectorizer(max_features=2000,
min_df=5, max_df=0.7, stop_words=stopwords.words('french'))
X = tfidfconverter.fit_transform(data["description"]).toarray()
```

Les données sont divisées en quatre parties : `X_train`, `X_test`, `y_train` et `y_test`. Le modèle est formé avec 80% des données (`X_train`, `y_train`) et testé avec les 20% restants (`X_test`, `y_test`). Cette configuration fractionnée est pivotée quatre fois pour une couverture complète de l'ensemble de données.

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=0)
```

Le code pour lancer l'algorithme de forêts aléatoires (Random Forest) est donné ci-dessous :

5. Exemples pour le traitement naturel du langage : https ://realpython.com/python-keras-text-classification/
6. Voir par exemple https ://python.gotrained.com/tf-idf-twitter-sentiment-analysis/. La communauté scientifique et les développeurs sont très présents dans les forums de Stack Overflow ou github

```
from sklearn.ensemble import RandomForestClassifier
text_classifier = RandomForestClassifier(n_estimators=100,
                                         random_state=0)
text_classifier.fit(X_train, y_train)
```

Le code pour l'algorithme SVM (avec un noyau linéaire) est donné par :

```
from sklearn.svm import SVC
text_classifier = SVC(kernel='linear')
text_classifier.fit(X_train, y_train)
```

Ici, c'est l'algorithme SVM avec un noyau linéaire mais utilisant la méthode de descente du gradient :

```
from sklearn.linear_model import SGDClassifier
text_classifier = SGDClassifier(loss='hinge', penalty='l2',
                          alpha=1e-3, n_iter=5, random_state=42)
text_classifier.fit(X_train, y_train)
```

Le code pour l'apprentissage via les réseaux de neurones artificiels est donné ci-dessous :

```
from sklearn.neural_network import MLPClassifier

text_classifier = MLPClassifier(activation='relu', alpha=1e-05,
              batch_size='auto', beta_1=0.9, beta_2=0.999,
              early_stopping=False, epsilon=1e-08,
              hidden_layer_sizes=(5, 10), learning_rate='constant',
              learning_rate_init=0.001, max_iter=200, momentum=0.9,
              n_iter_no_change=10, nesterovs_momentum=True, power_t=0.5,
              random_state=1, shuffle=True, solver='lbfgs', tol=0.0001,
              validation_fraction=0.1, verbose=False, warm_start=False)

text_classifier.fit(X_train, y_train)
score = text_classifier.score(X_test, y_test)
print("Accuracy:", score)
```

Le code pour l'algorithme Naive Bayes est :

```
from sklearn.naive_bayes import MultinomialNB
text_classifier = MultinomialNB()
text_classifier.fit(X_train, y_train)
```

B.3. Prévision

La commande `text_classifier.predict` permet de prédire la classe d'un nouveau texte.

```
predictions = text_classifier.predict(X_test)
```

Les différentes mesures de performances (Exactitude, Précision, Rappel, $F_1 score$, Matrice de confusion) sont données grâce au code suivant :

```
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score

print(confusion_matrix(y_test,predictions))
print(classification_report(y_test,predictions))
print(accuracy_score(y_test, predictions))
```

Ines Abdeljaoued Tej est Maître-Assistante au département Statistique de l'Ecole Supérieure de la Statistique et de l'Analyse de l'Information de Tunisie. Elle est chargée de l'enseignement de la programmation mathématique et plus particulièrement des méthodes d'apprentissage via les réseaux neuronaux. Elle est membre du laboratoire de BioInformatique, bioMathématique et bioStatistique de l'Institut Pasteur de Tunis. Elle est titulaire d'une maîtrise de Mathématiques pures à la Faculté des Sciences de Tunis, d'un DEA en Algorithmique à l'Ecole Polytechnique et d'une thèse de Calcul Formel à l'Université Pierre et Marie Curie, Paris VI, Sorbonne Université, France.