



Documentation

---

# keyBoost - Un méta-modèle par consensus guidé pour l'extraction de mots-clés

---

BEKKAR Zakaria  
zakaria.bekkar@ens-paris-saclay.fr

ECOLAB  
ENS PARIS-SACLAY - ENSAE PARIS

Juillet 2021

# Sommaire

<b>1</b>	<b>Introduction</b>	<b>2</b>
1.1	Contexte . . . . .	2
1.2	Brève revue de littérature dans le domaine de la <i>keyword extraction</i> . . . . .	3
1.3	<i>keyBoost</i> : Apports scientifiques et pratiques pour le cas d’usage avec les DREALs ( et tous les autres cas d’usages) . . . . .	4
<b>2</b>	<b>Fonctionnement de <i>keyBoost</i></b>	<b>5</b>
2.1	Vue générale du modèle . . . . .	5
2.2	Consensus guidé . . . . .	6
2.2.1	<i>Statistical Consensus</i> . . . . .	6
2.2.2	<i>Ranking Based Consensus</i> . . . . .	10
2.2.3	Orientation du consensus via le <i>feedback</i> utilisateur . . . . .	11
<b>3</b>	<b>Évaluation</b>	<b>12</b>
3.1	Validation de la pertinence de l’architecture <i>keyBoost</i> . . . . .	12
3.1.1	Méthodologie . . . . .	12
3.1.2	Résultats . . . . .	14
<b>4</b>	<b>Application web <i>keyBoost</i></b>	<b>15</b>
<b>5</b>	<b>Le package python <i>keyBoost</i></b>	<b>16</b>
5.1	A propos du package . . . . .	16
5.2	Installation . . . . .	16
5.3	Utilisation de base . . . . .	17
5.3.1	Modifier la longueur des <i>keywords</i> . . . . .	17
5.3.2	Modifier le nombre de <i>keywords</i> en sortie . . . . .	17
5.3.3	Inclure des <i>stopwords</i> . . . . .	18
5.3.4	Langues supportées . . . . .	18
5.4	Modèles sous-jacents . . . . .	19
5.5	Type de consensus . . . . .	19
5.5.1	Informations additionnelles pour le <i>statistical consensus</i> . . . . .	20
5.6	Modeles d’ <i>embedding</i> . . . . .	20

# 1 Introduction

## 1.1 Contexte

L'un des nombreux rôles des Directions régionales de l'Environnement, de l'Aménagement et du Logement (DREAL) se structure autour de leur statut particulier d'autorité environnementale. En cette qualité, elles sont notamment amenées à formuler des avis sur tout projet à impact environnemental sur leur territoire de compétence.

Une expérimentation conjointe menée par l'Ecolab, laboratoire d'innovation en intelligence artificielle (IA) affilié au Ministère de la Transition Ecologique, et la DREAL Bretagne explore les potentialités de l'apport de l'IA dans cette tâche par le biais de la production de *preuves de concept* (PoC).

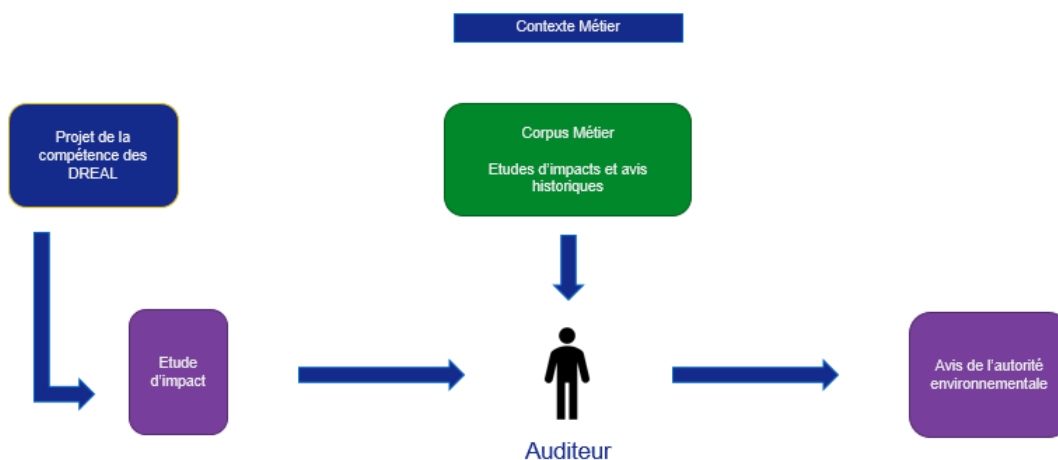


FIGURE 1.1 – Contexte Métier

Un des traits saillants du travail quotidien des auditeurs en charge de la formulation de ces avis est l'impératif de traiter une grande masse d'informations de façon à la fois fine et diligente : tout manquements aux délais impartis par le règlement européen en la matière pouvant s'assortir de lourdes sanctions financières. Le bénéfice de la mise en place d'outils agissant en support de la rédaction et de l'analyse documentaire est donc double, puisqu'il s'exprime tant du point de vue du gain financier que du point de vue de la charge de travail des auditeurs en leur offrant la possibilité de se concentrer sur ce qui xfait la véritable valeur ajoutée de leur expertise.

Ce constat à conduit à l'élaboration d'une *PoC* autour de la notion de *sommaire augmenté*. L'ambition affichée est de pouvoir permettre, par le biais d'un interface utilisateur, une extraction structurée et pertinente de l'information contenue dans les études d'impacts qui forment la base documentaire des avis de l'autorité environnementale. L'une des directions prises pour le *sommaire augmenté* a été la volonté de mettre en place un module d'extraction de mot-clés opérant sur les nombreuses sections composant les études d'impacts. A mon arrivée à l'Ecolab, j'ai hérité de la responsabilité de concevoir et d'implémenter ce module en parallèle de l'élaboration d'une *PoC* cette fois-ci sur un système de recommandation d'avis.

## 1.2 Brève revue de littérature dans le domaine de la *keyword extraction*

L'extraction de mots clés (également appelée détection de mots clés/ analyse de mots clés/ keyword extraction) fait référence à l'ensemble des techniques d'analyse de texte utilisées pour extraire automatiquement les mots et expressions les plus importantes d'un texte. Elles permettent de reconnaître les principaux sujets abordés et de résumer le contenu d'un texte de manière succincte.

A ce titre, c'est un sous-domaine du traitement naturel du langage (TLN ou NLP) qui s'est peu à peu consolidé autour de deux axes : *Machine Learning* vs Methodes Statistiques vs Théorie des Graphes et Modèles Supervisés vs Modèles Non-Supervisés.

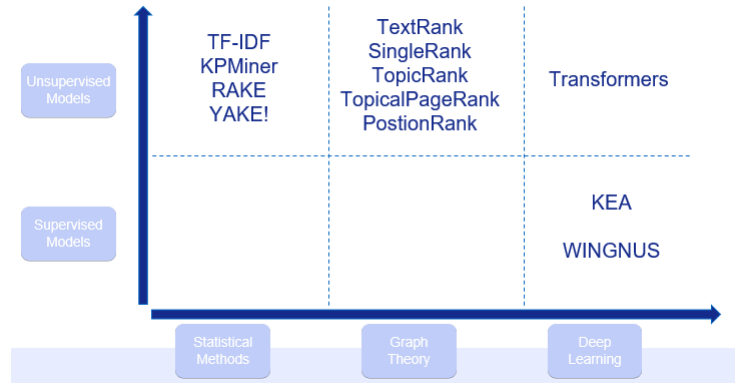


FIGURE 1.2 – Périmètre du sous-domaine scientifique de la keyword extraction

La pléthore de directions dans laquelle la recherche se structure reflète une réalité à laquelle on peut difficilement échapper lorsque l'on fait de la keyword extraction : il n'existe pas d'approche univoque surpassant toutes les autres dans tous les scénarios. Les différentes typologies de modèles sont performantes sur différents types de textes. [1]

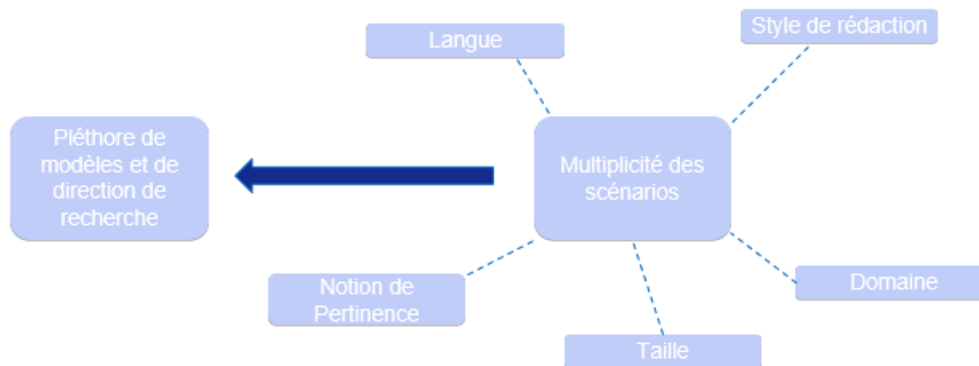


FIGURE 1.3 – La difficulté d'avoir un modèle uniformément le plus pertinent.

Malgré des efforts de recherches considérables consacrés à ce sujet au fil des ans, le problème de l'extraction de mots-clés pertinents avec une grande précision reste non résolu [2]. Tout un ensemble de facteurs sont en cause. Parmi ceux-ci, les difficultés à définir clairement la notion de pertinence ou encore la large diversité linguistique à laquelle les algorithmes sont confrontés (langues, taille, styles de rédaction, domaines etc) sont parmi les défis les plus brûlants.

D'autres obstacles tiennent aux problèmes posé par les mots-clés absents, la restriction de correspondance exacte entre mots-clés labellisés et prédits et le nombre élevé de mots-clés candidats qui peuvent être générés à partir d'un seul texte. Toutes ces questions mettent en évidence les freins au développement d'une solution globale et motivent la nécessité de poursuivre les recherches.

### 1.3 *keyBoost* : Apports scientifiques et pratiques pour le cas d'usage avec les DREALs ( et tous les autres cas d'usages)

Dans ce contexte, je propose un modèle intitulé *keyBoost* qui est conçu pour palier le problème du choix du modèle de *keyword extraction* en particulier pour les cas d'usages où :

- il n'y a ni données labellisées
- ni recul sur les performances potentielles de chaque typologie de modèle

Les données de la DREAL Bretagne, étant de cette nature, offrent un cadre d'application parfait pour cette architecture originale dans la littérature du domaine de la *keyword extraction*

*keyBoost* se définit en première lieu par son architecture de *méta-modèle par consensus guidé*. Concrètement, le modèle se compose d'une sélection de modèles considérés *state of the art* pour leur typologie et fait en sorte, via un système de consensus qui sera explicité ci-après, de mettre en cohérence l'ensemble des keywords générées par ces modèles pour ne proposer que ceux qui sont globalement les plus susceptibles d'être pertinents.

Les 3 principaux sous-modèles sur lesquels *keyBoost* est basé sont YAKE! [3], KeyBERT [4] et TextRank [5]. Ce choix reflète, outre les performances et les excellentes implémentations disponibles, la volonté de diversifier les typologies des techniques utilisées (*statistiques* pour YAKE!, *Deep learning* pour KeyBert et *Graphs* pour TextRank). Cette variété est pensée comme permettant de s'adapter au maximum cas possibles.

Cette approche, confrontée à l'expérimentation permet, en l'absence d'a priori sur le meilleur modèle à utiliser, d'avoir au moins la validation du critère du *maximin* et au mieux une performance plus élevée que la moyenne des sous-modèles utilisés. Résultats qui assurent l'intérêt et la portée opérationnelle de cette proposition.

## 2 Fonctionnement de *keyBoost*

### 2.1 Vue générale du modèle

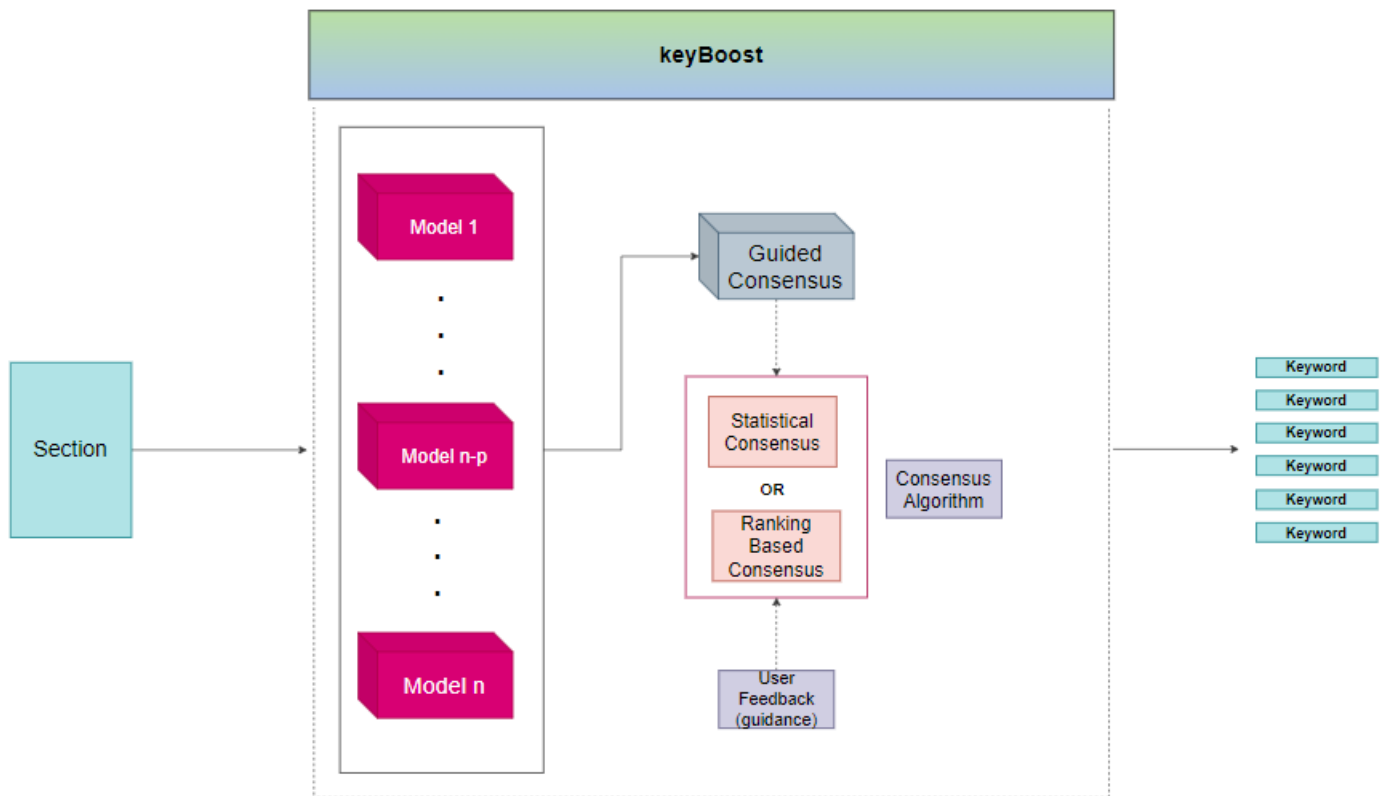


FIGURE 2.1 – Vue d'ensemble de *keyBoost*

Comme mis en évidence sur le schéma ci-dessus, *keyBoost* se fonde sur les mots-clés générés par un ensemble de modèles triés sur le volet et en sélectionne les plus pertinents via un *consensus guidé* qui peut s'apparenter intuitivement à un vote.

Le coeur de l'architecture se situe, par voie de conséquence, dans le consensus guidé qui permet d'harmoniser et de juger l'ensemble des sorties des sous-modèles retenues. L'adjectif "guidé" ici à toute son importance puisque ce n'est pas simplement un processus de choix algorithmique déconnecté du contexte de son cas d'usage. Il semblait important de permettre au modèle de prendre en compte le jugement des utilisateurs sur les mots-clés proposés afin de "biaiser" consensus des prochaines extraction sur les modèles les plus pertinents.

Dans le cadre du développement de *keyBoost*, deux approches ont été explorées parmi les nombreuses possibilités envisageables : une consensus statistique et un consensus basé sur le rang.

## 2.2 Consensus guidé

### 2.2.1 Statistical Consensus

L'idée du *statistical consensus* est d'exploiter les scores en sorties de tous les modèles *shortlistés* afin d'avoir une base de comparaison et de sélection. L'obstacle majeur à l'application directe de cette intuition est le fait que ces scores ne sont en réalité pas cohérents les uns avec les autres. Les procédés de génération des scores sont très différents les uns des autres en fonction des architectures utilisées sans parler du sens même que ces quantités peuvent avoir ( par exemple la similarité calculée via le modèle de *deep learning* BERT ou bien encore le score purement statistique en sortie de *TF-IDF*)

Le *statistical consensus* dépasse cette impossibilité de recouvrer toutes interprétations cardinales globales des scores en tentant de reconstruire statistiquement une *cardinalité approchée*. Cette dernière est largement suffisante pour pouvoir opérer une sélection pertinente des mots-clés.

Le procédé retenu est le suivant :

1. Sélection de modèles statistiques ayant la particularité d'être discriminant (DSM). Ce caractère discriminant est fondamental, il fait référence au fait que nous voulons quelques mots-clés très bien notés qui se séparent du reste dans une sorte de structure pyramidale : la majorité des mots-clés candidats ont des scores faibles/médiocres, plus le score est élevé, moins il y a de mots-clés candidats. La selection des DSM se fait au regard du couple densité de probabilité et paramètre exhibant :
  - (a) *skewness positive* élevée
  - (b) *kurtosis* faible

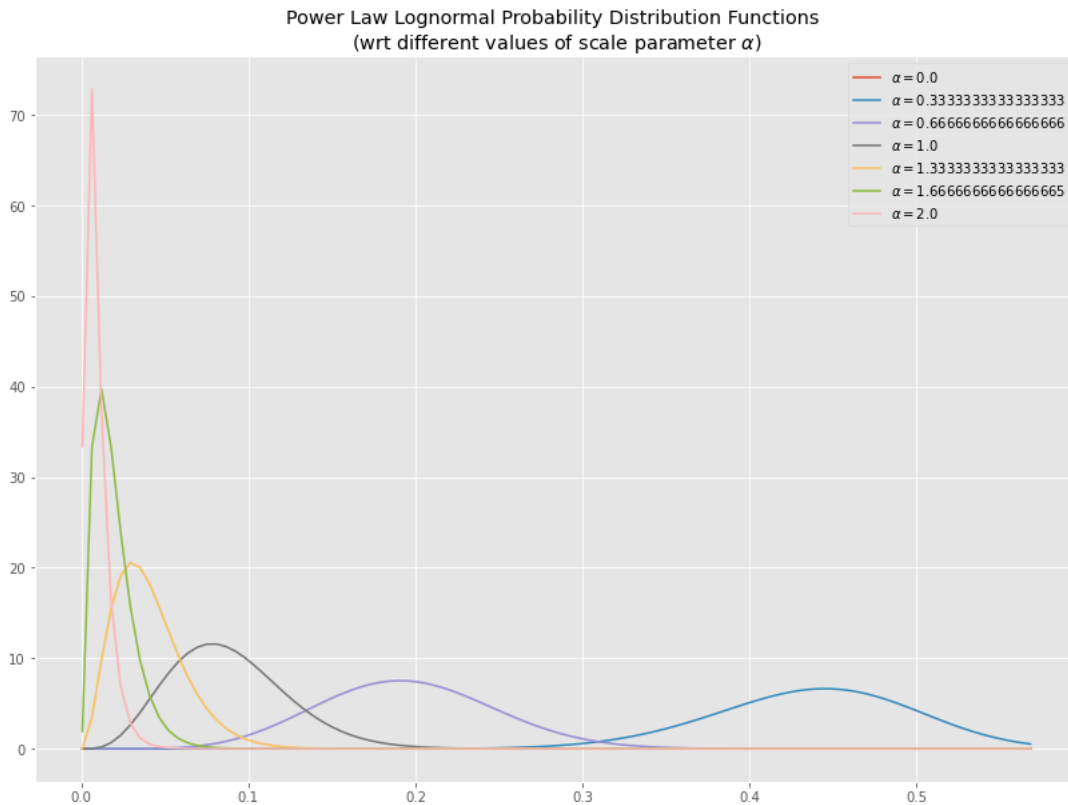


FIGURE 2.2 – Densités de lois de puissance lognormale pour différente valeur du paramètre  $\alpha$

La figure ci-dessus illustre cette double articulation entre modèle statistique prompt à la discrimination, ce qui est le cas pour la loi de puissance lognormale, et la bonne plage du paramètre. (ici  $0 < \alpha < 1$ ) permettant d'obtenir un DSM).

Ce travail de sélection a débouché sur une *pool* de quatre DSM :

- (a) Loi de puissance
  - (b) Gamma
  - (c) Pareto
  - (d) Loi de puissance lognormale
2. Appliquer une transformation à tous les scores pour les faire vivre dans le même espace. Plusieurs choix sont possibles, la simplicité a été privilégiée en optant pour la normalisation ou la normalisation inverse selon les cas.

$$S_k^* = \frac{s_k - \frac{1}{n} \sum_{k=1}^n s_k}{\frac{1}{n} \sum_{k=1}^n (s_k - \frac{1}{n} \sum_{k=1}^n s_k)^2} \quad (2.1)$$

$$\overline{S}_k^* = \frac{\frac{1}{s_k} - \frac{1}{n} \sum_{k=1}^n \frac{1}{s_k}}{\frac{1}{n} \sum_{k=1}^n (\frac{1}{s_k} - \frac{1}{n} \sum_{k=1}^n \frac{1}{s_k})^2} \quad (2.2)$$

3. Détermination du DSM le plus approprié aux keywords et aux scores associés en sortie de chaque sous-modèle. Ceci est effectué avec un algorithme d'adéquation (*fitting*) qui suit le programme d'optimisation suivant :

$$\mathbb{P}_\theta^* = \arg \min_{\mathbb{P}_\theta \in \Psi} \|\hat{f} - f_\theta\|_2^2$$

avec  $\Psi :=$  Ensemble DSM

$\hat{f}$  : densité estimée *via* kernel density estimation (KDE)

$f_\theta$  : densité associée à  $\mathbb{P}_\theta$

(2.3)

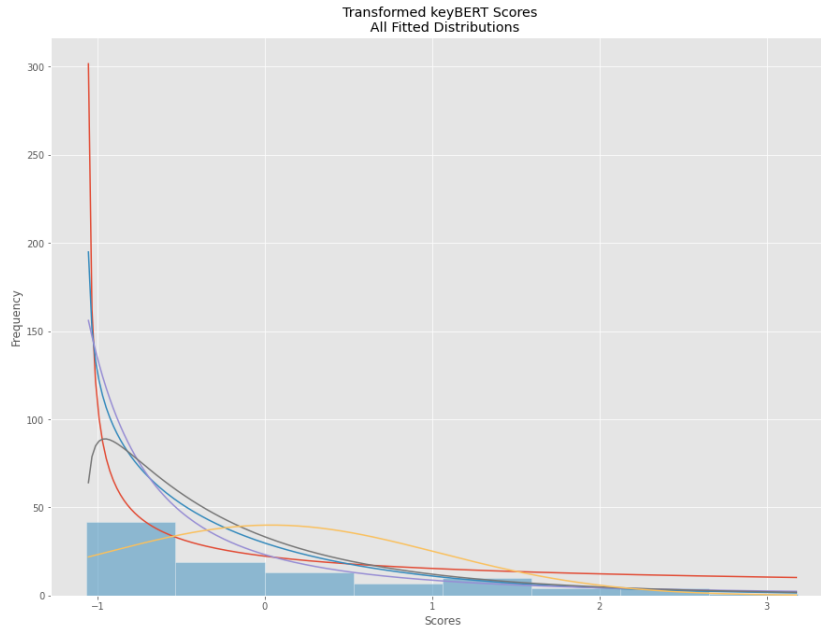


FIGURE 2.3 – Illustration graphique de l'étape de sélection du DSM

Comme le montre la figure 2.3, la sélection du DSM revient graphiquement à choisir la courbe de densité la plus "proche" de l'histogramme des scores.



4. Dans le cadre du DSM le plus approprié, on effectue une estimation du paramètre le plus en phase avec les données en présence à l'aide du maximum de vraisemblance (MLE). Si ce paramètre est cohérent avec la région paramétrique définie en étape 1 alors on poursuit l'algorithme de *consensus statistique* sinon on bascule sur le *ranking consensus*. L'estimation du paramètre par MLE se fait via le programme :

$$\hat{\theta} = \arg \max_{\theta \in \Theta} \prod_{k=1}^n f_{\theta}(S_k^*) \quad (2.4)$$

5. Le couple (DSM,paramètre estimé) est utilisé comme *a priori* sur le véritable processus générateur de la distribution des scores. On effectue alors un test Kolmogorov-Smirnov au seuil de confiance de  $\alpha\%$  pour décider si cette hypothèse est vraisemblable au vu de la structure de nos données. Si tel est le cas, on passe à l'étape suivante de l'algorithme du *statistical consensus*, sinon on bascule sur le *ranking consensus*.
6. Application d'un algorithme de déduplication afin d'éliminer les doublons au sens strict et/ou sémantique parmi les keywords qui sont proposés au niveau global. Cet algorithme est assez sophistiqué dans le sens où il utilise un *embedding* des keywords issue d'une architecture Transformers. Cet embedding est ensuite exploité pour éliminer les doublons, c'est à dire les keywords dont la représentation vectorielle est trop proche. Cette notion de proximité est calculé en tirant parti de la *similarité cosinus*.
7. Sélection des *top k-keywords* selon le score transformé

En page suivante se trouve le diagramme des flux du processus du *Statistical Consensus* qui reprend graphiquement l'ensemble des étapes évoquées ci-dessus.

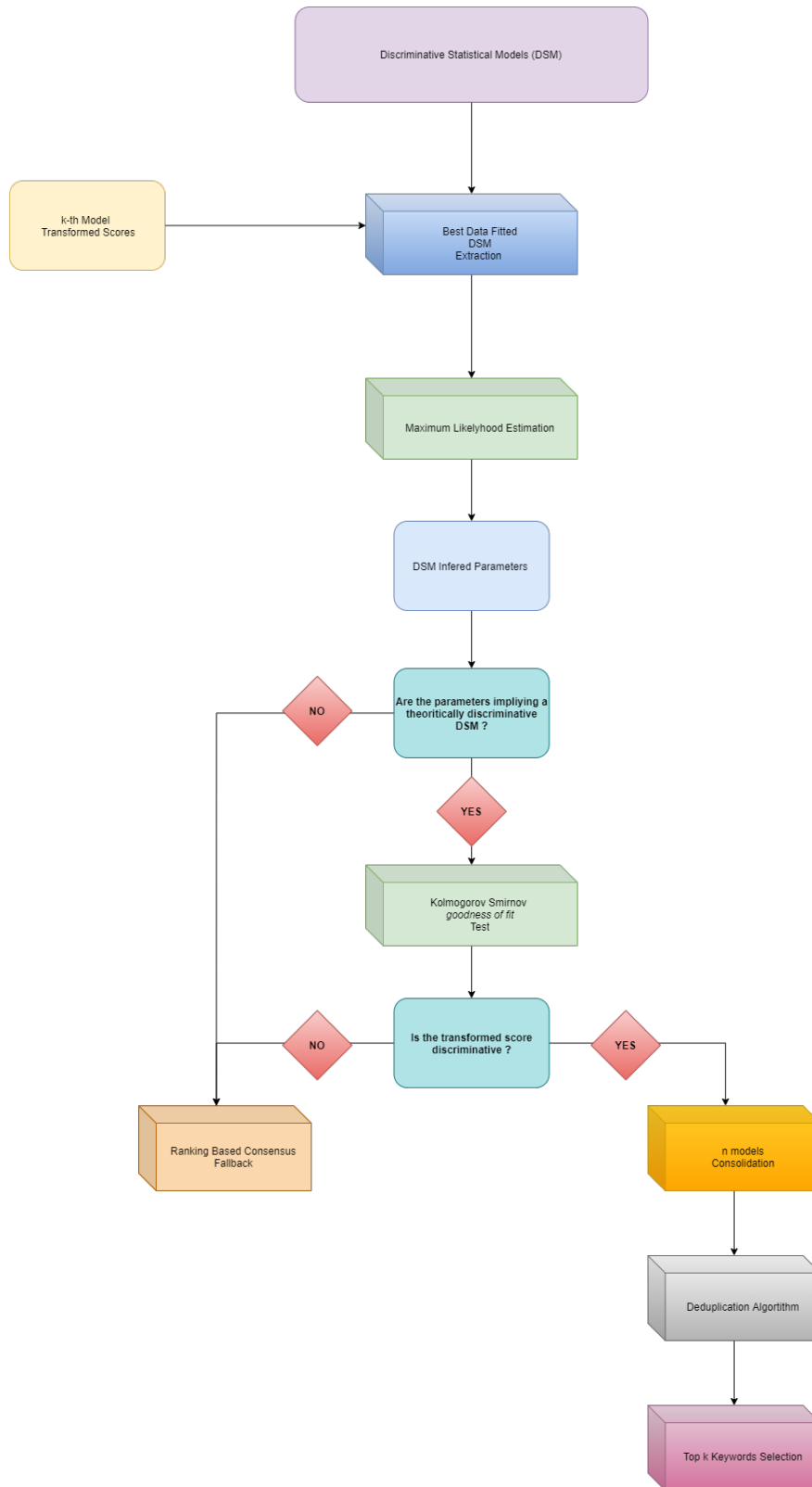


FIGURE 2.4 – Diagramme des flux du processus du *Statistical Consensus*

### 2.2.2 Ranking Based Consensus

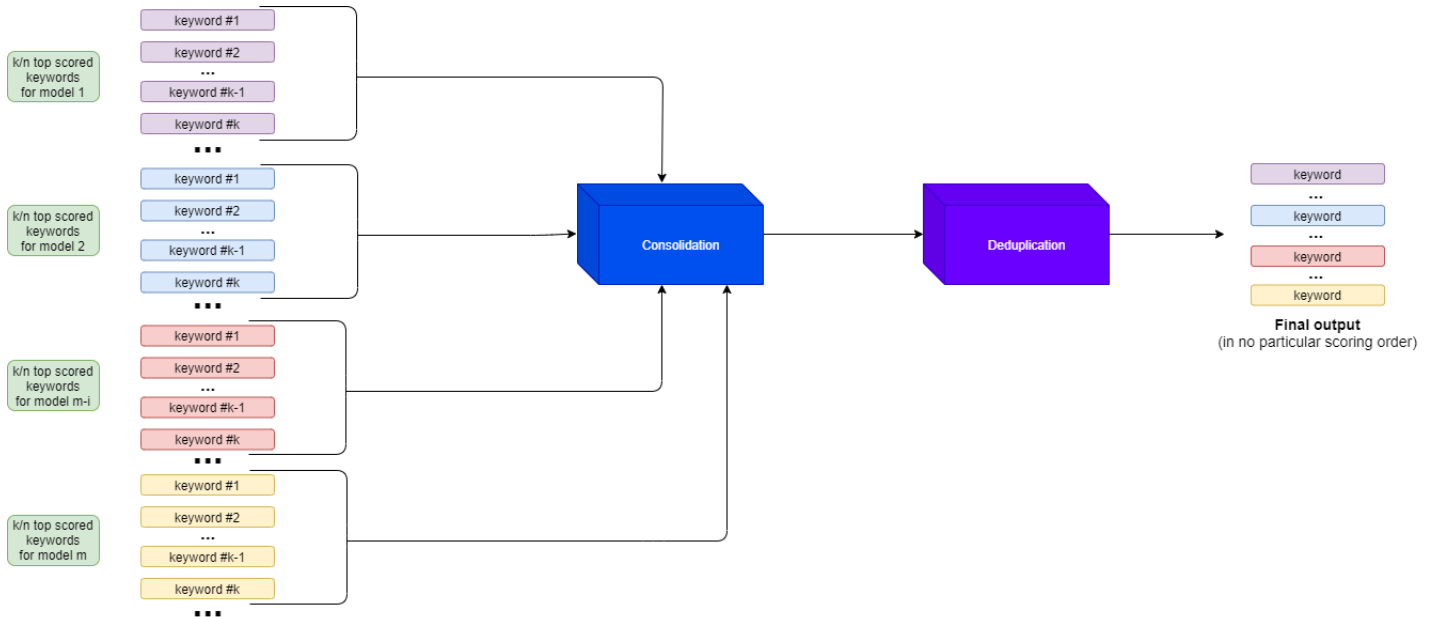


FIGURE 2.5 – Diagramme des flux du processus du *Ranking Based Consensus*

Toujours dans cette recherche d’une interprétation globale la plus fidèle possible de la pertinence de la proposition de keyword de chaque sous-modèle, l’approche *ranking based* se fonde non plus sur les scores (*cardinalité*) mais directement sur les classements (*ordinalité*).

Selon un nombre  $k$  de keywords voulus et  $n$  de modèles sous-jacents, la procédure est comme suit :

1. Sélection des  $E(\frac{k}{n})$  keywords les plus pertinents selon le score de chaque modèle
2. Application de l’algorithme de déduplication

Une limite de cette algorithme par rapport à l’approche statistique est son caractère rustre. En effet, par construction le *ranking based consensus* ignore les potentielles proposition à partir du  $E(\frac{k}{n}) + 1$  rang d’un sous-modèle plus pertinentes que les  $E(\frac{k}{n})$  premières d’un autre sous-modèle ( avec  $E(.)$  fonction *partie entière*).

Cependant, en pratique, c’est cette même simplicité qui caractérise ce consensus qui lui confère toute son efficacité. Son approche simple de sélection de meilleures prédiction pour chaque modèle peut même s’avérer dans certains cas supérieure au *statistical consensus* malgré les limites qui viennent d’être évoquées, du fait notamment que l’on reste sur de l’ordinalité pure.

### 2.2.3 Orientation du consensus via le *feedback* utilisateur

L'idée est d'adjoindre à la procédure de consensus purement algorithmique de l'information issue du domaine via le *feedback utilisateur*, le consensus devient alors *guidé*. C'est fondamentalement de l'*active learning* : un procédé par lequel on essaie d'améliorer les propositions du méta-modèle en tirant parti des retours éclairés de l'utilisateur. Par ce biais, un cercle vertueux est créé de bonification de l'algorithme avec le temps, s'approchant de plus en plus de ce qui est recherché par les usagers.

Concrètement dans le cas de *keyBoost*, un feedback se manifeste par un jugement de l'utilisateur final de la *keyword extraction* sur la qualité des propositions. Cette information est ensuite utilisée pour biaiser le consensus en faveur des modèles les plus adaptés au cas d'usage et en l'occurrence ceux jugés les plus pertinents. Les prochains consensus pour le même type de texte mettront en avant ces modèles ce qui se traduira par une sur-représentation de leurs keywords.

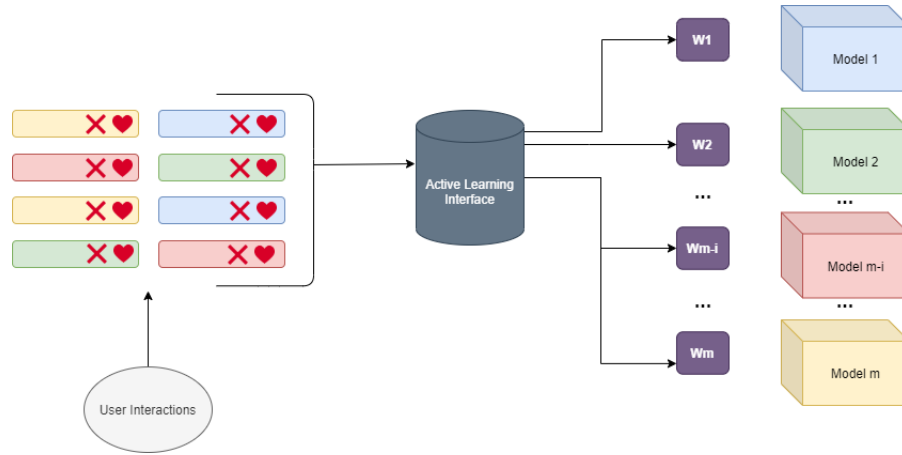


FIGURE 2.6 – Diagramme du *feedback utilisateur*

La figure ci-dessus retrace techniquement le fonctionnement du feedback utilisateur, essentiellement l'idée est de faire interagir l'utilisateur avec les sorties par le biais de deux boutons indiquant l'adéquation ou l'inadéquation de la sortie obtenue. Ces retours sont alors utilisés pour noter les sous-modèles responsables de ces propositions. Pour les prochaines interrogations de *keyBoost* dans le même contexte, les sous-modèles les mieux notés seront ceux qui auront le plus de poids dans le consensus. En définitive, le *feedback utilisateur* modifie le poids de chaque sous-modèle dans le consensus afin de le *guider* au plus près des attentes de l'utilisateur.

Il est à noter que cette fonctionnalité ne sera pour l'instant pas disponible au sein du package *keyBoost* car très dépendante des choix d'architecture d'un projet.

## 3 Évaluation

### 3.1 Validation de la pertinence de l'architecture keyBoost

#### 3.1.1 Méthodologie

Le point de départ de la méthodologie de validation de l'apport effectif de *keyBoost* est la volonté de mettre en évidence qu'en l'absence d'a priori sur le meilleur modèle à utiliser pour un domaine et un type de texte spécifique, *keyBoost* démontre de bonnes capacités :

1. Soit de part une pertinence plus élevée que le modèle ayant la performance la plus faible parmi les sous-modèles employés, i.e critère du *maximin*
2. Soit de part une pertinence plus élevée que la moyenne des sous-modèles utilisés.

A noter qu'ici la notion de moyenne est celle de la moyenne arithmétique *naïve*, il peut être en effet tentieux de considérer que la moyenne de la performance de modèles sur un ensemble de données de test correspond à la moyenne arithmétique de leur performance respectives sur le ce même jeu de données. Cette construction est utilisée avant tout de manière indicative et vise à illustrer le potentiel de *keyBoost* dans certains contextes.

Fort de cette approche et dans l'esprit tant de la reproductibilité que la significativité des résultats en sortie de cette expérimentation, nous confrontons *keyBoost* à un dataset classique de *benchmarking* au sein de la littérature scientifique de l'extraction de mots-clefs : PubMed.

L'ensemble de données choisi est basé sur des articles collectés dans leur intégralité auprès de PubMed Central, qui comprend plus de 26 millions de citations pour la littérature biomédicale, des revues de sciences de la vie et des livres en ligne. En particulier, notre *dataset* se compose de 500 articles de sciences médicales sélectionnés à partir cette même source. Chaque article est associé avec un nombre variable de mots-clés déterminés manuellement par des experts du domaine médical.

Ce *dataset* est réputé extrêmement difficile en raison du caractère très spécifique du vocabulaire utilisé débouchant sur des performances très faibles des extracteurs de mots-clés quel que soit l'approche utilisée (*statistique, deep learning, machine learning ou graphes*.) C'est en partie ce qui a motivé le choix de ces données afin d'opposer directement *keyBoost* au contexte le plus difficile et de juger au mieux de ses performances réelles pour potentiellement tous les cas d'usages.

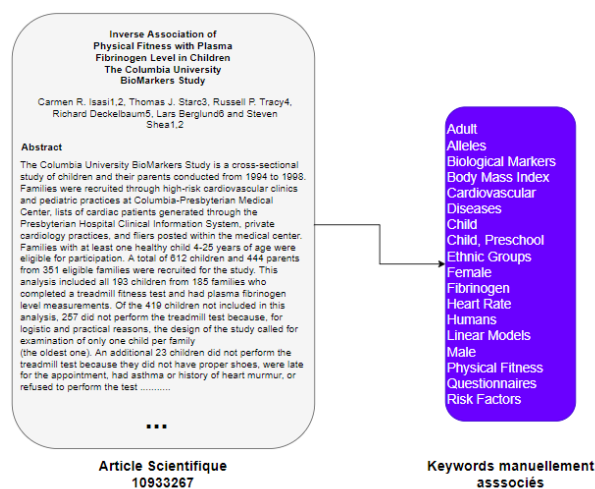


FIGURE 3.1 – Exemple d'une association article scientifique et keywords présente au sein de PubMed

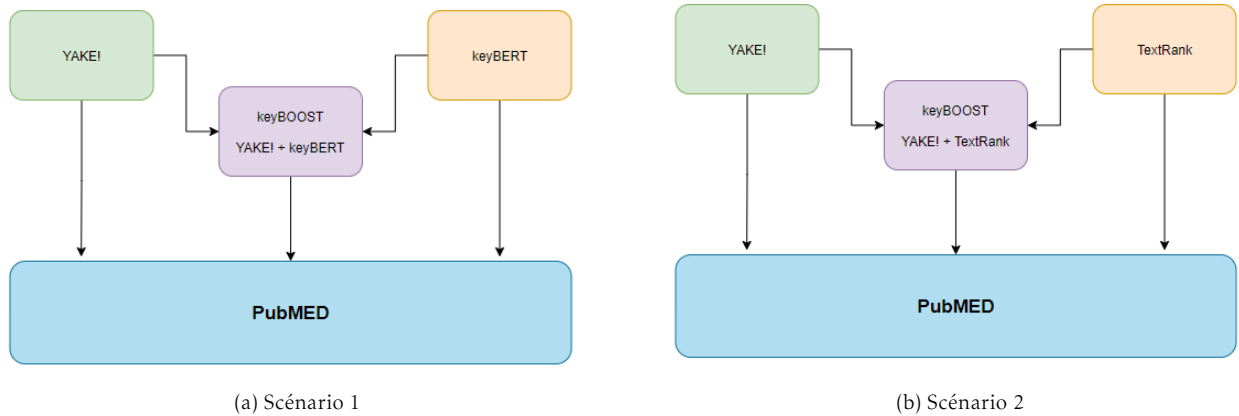


FIGURE 3.2 – Méthodologie d'évaluation

Plus précisément, deux scénarios d'évaluation ont été choisis :

1. Scénario 1 : on évalue YAKE! et keyBERT séparément sur PubMed et on confronte ces performances à celle de keyBoost avec les sous-modèles YAKE! et keyBERT.
2. Scénario 2 : on évalue YAKE! et TextRank séparément sur PubMed et on confronte ces performances à celle de keyBoost avec les sous-modèles YAKE! et TextRank.

Le choix de YAKE! comme pivot présent dans les deux scénarios n'est pas anodin puisqu'il démontre d'assez bonnes performances malgré la difficulté de l'extraction de mots-clés sur ce corpus. En effet, la plupart des autres modèles ont une correspondance quasiment nulle avec les keywords sélectionnés par les experts selon la métrique utilisée.

En parallèle, keyBERT a été explicitement choisi dans l'un des scénarios afin de mettre en difficulté keyBoost : l'architecture de *Transformers* sur lequel ce modèle est fondée ne calcule pas explicitement un score issu d'un algorithme quelconque mais bien une similarité entre l'embedding des mots-clés candidats et l'embedding moyen du texte. En fonction de la nature de ces embeddings et du type de texte, le consensus statistique ou bien même du rang peut avoir quelques difficultés à départager les meilleurs candidats contrairement à ce qui peut se passer pour les méthodes plus classiques (*statistique, machine learning ou graphes* et qui composent la majorité des extracteurs aujourd'hui disponibles.)

Nous ne testons que deux modèles par scénario essentiellement pour des considérations de puissance de calculs disponible. Toutefois, il est bon de signaler que *keyBoost* n'implique aucunement un surplus de complexité algorithmique : sa complexité correspond simplement à la somme des complexités des sous-modèles.

Pour notre évaluation, nous utiliserons la métrique *F1@10 macro average* qui correspond au score F1 des 10 premiers keywords proposés par les modèles utilisés. La mention *macro average* fait référence au fait que ce score est globalisé sur l'ensemble des prédictions faites le dataset en prenant simplement la moyenne arithmétique des *F1@10* pour chaque association article-keywords.

### 3.1.2 Résultats

YAKE!	keyBERT	keyBOOST (YAKE! + keyBERT)	
0.0279	0.0	0.0089 Statistical Consensus	0.0 Rank Consensus

(a) Scénario 1

YAKE!	TextRank	keyBOOST (YAKE! + TextRank)	
0.0279	0.0	0.0112 Statistical Consensus	0.0187 Rank Consensus

(b) Scénario 2

FIGURE 3.3 – Résultat de l'évaluation sur PubMed ( $F1@10$  macro average)

Sur les deux cas de figure, *keyBoost* s'aligne sur le critère du *maximin* évoqué dans la section méthodologie : il y'a toujours un type de consensus qui débouche sur une performance meilleure que celle que le pire modèle évalué isolément. De plus, on remarque que pour le scénario 2, les deux types de consensus apportent des résultats intéressants, le *rank consensus* affichant même un résultat supérieur à la moyenne de ceux de YAKE! et TextRank : i.e  $0.187 > 0.135$ .

Ces résultats démontrent la pertinence de *keyBoost* en l'absence d'a priori en terme de choix de modèle pour un domaine et un type de texte spécifique. Cette conclusion est à nuancer par le fait que pour des sous-modèles de type *Deep Learning*, il est de rigueur de rester vigilant notamment :

1. en diversifiant la *pool* de sous-modèles en y spécifiant une majorité de modèles de type *statistique, machine learning ou graphes*
2. en privilégiant le *statistical consensus*.

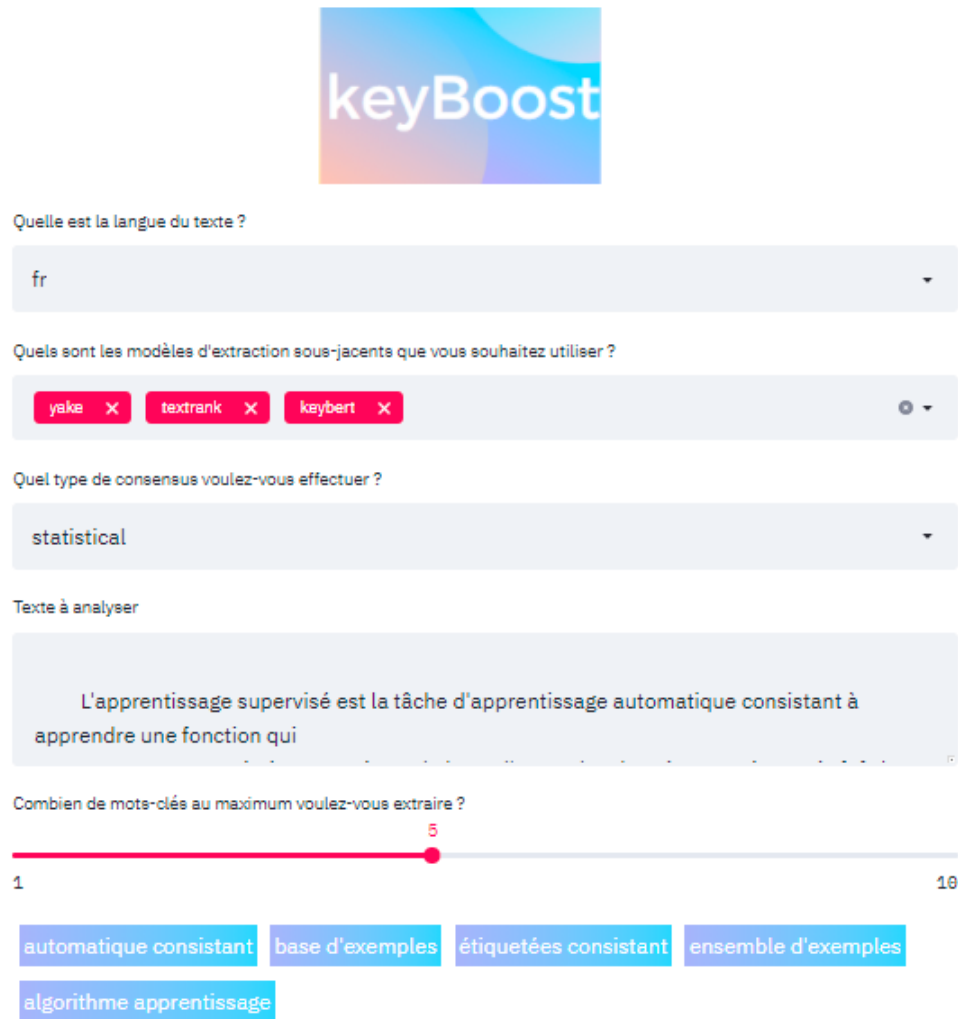
La proposition de *keyBoost* saura se montrer à la fois pertinente et utile pour toute situation où il y'a nécessité de traiter de gros volumes d'informations textuels dont on veut avoir une idée générale en un coup d'oeil.

L'extraction de mots-clés sans la condition de données préalablement labellisées ni d'expertise dans la sélection des modèles ouvre la possibilité d'application clé-en-main de *keyBoost* à tous les champs de l'administration et de l'industrie reconstruant des problématiques similaires.

Dans cet esprit, une *application web* permettant de tester à la volée les extractions de *keyBoost* est mise à disposition du public dans une optique pédagogique de familiarisation avec les potentialités de l'IA.

Les impératifs opérationnels quant eux sont également pris en compte par ce travail : tout service ou organisation ayant l'intention d'intégrer le module *keyBoost* au sein d'un projet pourra exploiter le package python du même nom et obtenir des extractions de mots-clés en quelques lignes de codes seulement. Les deux parties suivantes précisent ces deux aspects.

## 4 Application web *keyBoost*



The screenshot shows the keyBoost web application interface. At the top is the keyBoost logo. Below it are several configuration options:

- Quelle est la langue du texte ?**: A dropdown menu with 'fr' selected.
- Quels sont les modèles d'extraction sous-jacents que vous souhaitez utiliser ?**: A row of three red buttons labeled 'yake', 'textrank', and 'keybert', each with a close icon. A plus icon and a dropdown arrow are on the right.
- Quel type de consensus voulez-vous effectuer ?**: A dropdown menu with 'statistical' selected.
- Texte à analyser**: A large text area containing the text: 'L'apprentissage supervisé est la tâche d'apprentissage automatique consistant à apprendre une fonction qui'.
- Combien de mots-clés au maximum voulez-vous extraire ?**: A slider ranging from 1 to 10, with a red dot at 5.
- Buttons**: A row of five buttons: 'automatique consistant', 'base d'exemples', 'étiquetées consistant', 'ensemble d'exemples', and 'algorithme apprentissage'.

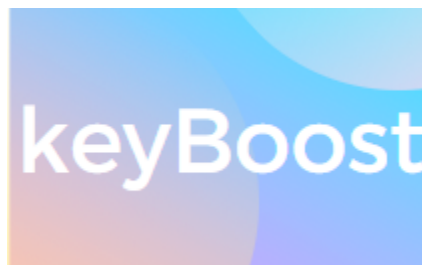
FIGURE 4.1 – Application Web

L'application web de démonstration des capacités de *keyBoost* est accessible [ici](#). Cette application est conçue pour permettre l'extraction de *keywords* à la volée sur le texte saisi par l'utilisateur. Elle permet également d'agir sur un ensemble de paramètres afférant à *keyBoost* :

1. langue du texte : choix entre anglais et français ( théoriquement les modèles sous-jacent sont capables de supporter avec une qualité éparse l'ensemble des langage à alphabet latin)
2. choix des modèles sous-jacents : selection d'un ou plusieurs modèles
3. choix du type de consensus : *statistical* ou *ranking*
4. champ de saisie du texte
5. choix du nombre de *keywords* à extraire (limité à 10 pour éviter les sorties médiocres de milieu de distribution)



## 5 Le package python *keyBoost*



### 5.1 A propos du package

Le package *keyBoost* est un outil d'extraction de mots clés simple et facile à utiliser qui évite d'avoir à sélectionner les meilleurs modèles pour votre cas d'utilisation spécifique. Aucune connaissance de la littérature sur l'extraction de mots-clés ou d'expertise n'est nécessaire pour extraire les meilleurs mots-clés possibles, c'est à dire sans aucunes connaissances préalables des modèles les plus performants pour votre tâche. Cette extraction est pensée pour être faisable en quelques lignes de code.

Dans la suite, nous allons explorer comment extraire des mots-clés avec *keyBoost* dans le scénario le plus classique. Ce faisant, nous présenterons les capacités de ce package et les paramètres que vous voudrez potentiellement adapter pour votre propre cas d'usage. Une documentation interactive sous forme de *Colab notebook* est disponible [ici](#), en plus du code source directement accessible sur [ce dépôt github](#)

### 5.2 Installation

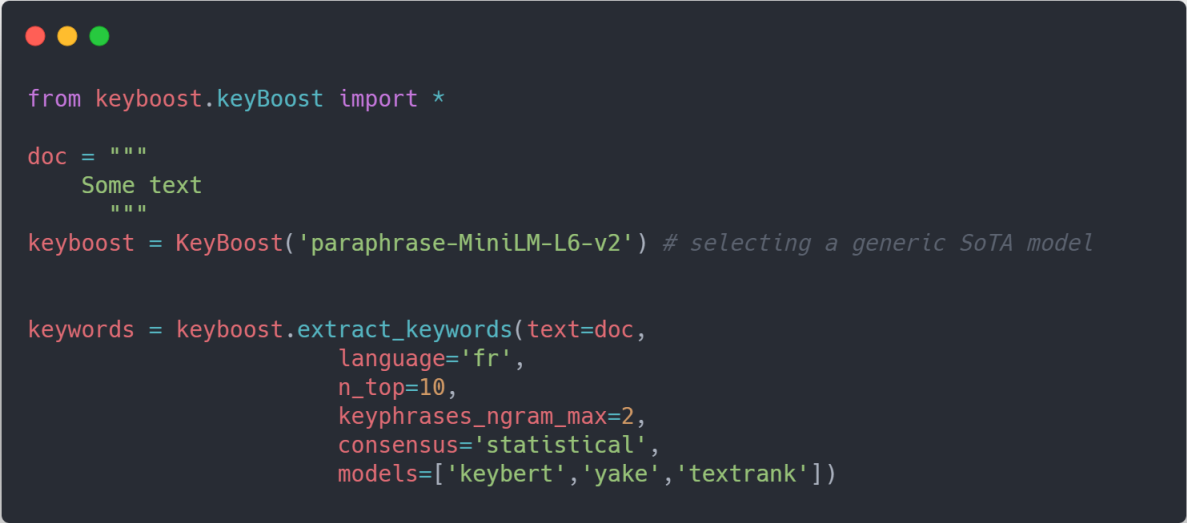
Une configuration préalable de python est requise pour faire fonctionner ce package. L'installation *keyBoost* s'effectue en exécutant cette ligne de commande au sein du terminal de votre machine :



FIGURE 5.1 – Installation

## 5.3 Utilisation de base

La tâche basique d'extraction des mots-clés d'un document peut se faire en quelques lignes de code :



```
from keyboost.keyBoost import *

doc = """
    Some text
    """

keyboost = KeyBoost('paraphrase-MiniLM-L6-v2') # selecting a generic SoTA model

keywords = keyboost.extract_keywords(text=doc,
                                     language='fr',
                                     n_top=10,
                                     keyphrases_ngram_max=2,
                                     consensus='statistical',
                                     models=['keybert', 'yake', 'textrank'])
```

FIGURE 5.2 – Installation

### 5.3.1 Modifier la longueur des *keywords*

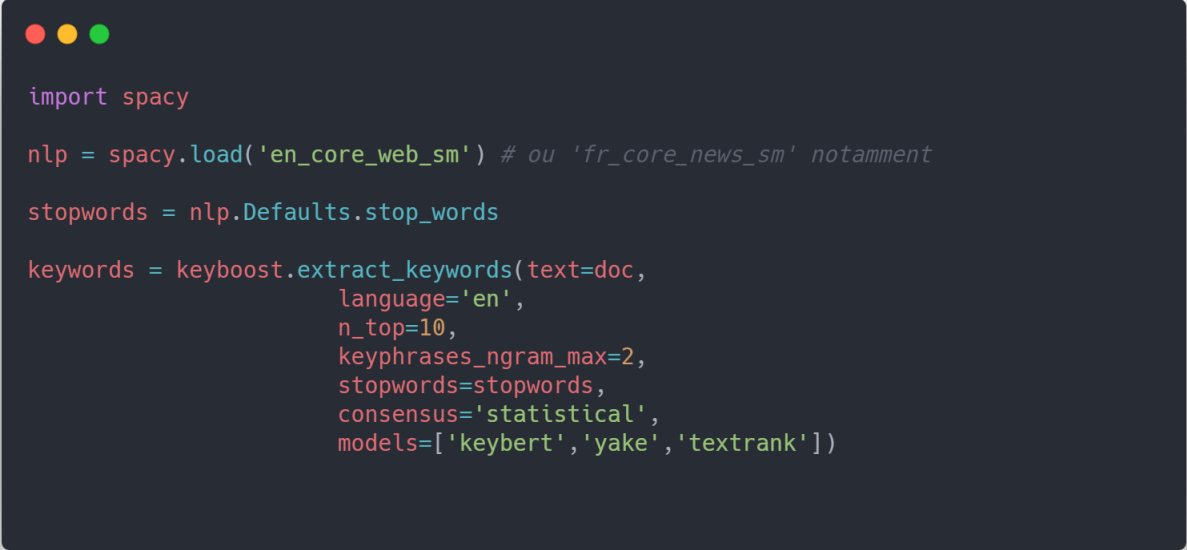
Pour ce faire, vous pouvez simplement définir l'argument *keyphrase\_n\_gram\_max* sur la limite supérieure de votre choix.

### 5.3.2 Modifier le nombre de *keywords* en sortie

Vous pouvez définir directement une valeur de *n\_top* qui correspond à ce que vous recherchez. Attention à ne pas fixer cet argument trop haut afin d'éviter d'avoir des extractions de qualités médiocres issues du coeur de distribution.

### 5.3.3 Inclure des *stopwords*

Pour de meilleurs résultats, c'est une bonne pratique d'inclure systématiquement des *stopwords*, une liste générique est largement suffisante en général. Cependant, une liste organisée et ciblée sur vos besoins ajoutera autant de pertinence aux extractions.



```
import spacy

nlp = spacy.load('en_core_web_sm') # ou 'fr_core_news_sm' notamment

stopwords = nlp.Defaults.stop_words

keywords = keyboost.extract_keywords(text=doc,
                                     language='en',
                                     n_top=10,
                                     keyphrases_ngram_max=2,
                                     stopwords=stopwords,
                                     consensus='statistical',
                                     models=['keybert', 'yake', 'textrank'])
```


FIGURE 5.3 – Ajouter des *stopwords*

### 5.3.4 Langues supportées

La modification du paramètre *language* avec le [code ISO 639-1](#) de la langue du document dont vous souhaitez extraire les mots-clés peut être utile. Le langage spécifique de votre document peut ne pas être pris en charge par les modèles d'extraction de mots clés sous-jacents ou par le modèle d'*embeddings*. Pour ceux qui sont principalement utilisés dans la littérature sur l'extraction de mots-clés, il n'y aura pas de problèmes.

## 5.4 Modèles sous-jacents

Par défaut, il est recommandé d'utiliser les 3 modèles supportés par keyBoost pour le moment (KeyBERT, YAKE! et TextRank). Si vous disposez d'informations vous amenant à ne vouloir utiliser qu'un sous-ensemble de ces modèles, vous pouvez simplement spécifier cette sous-liste comme valeur pour *models*.



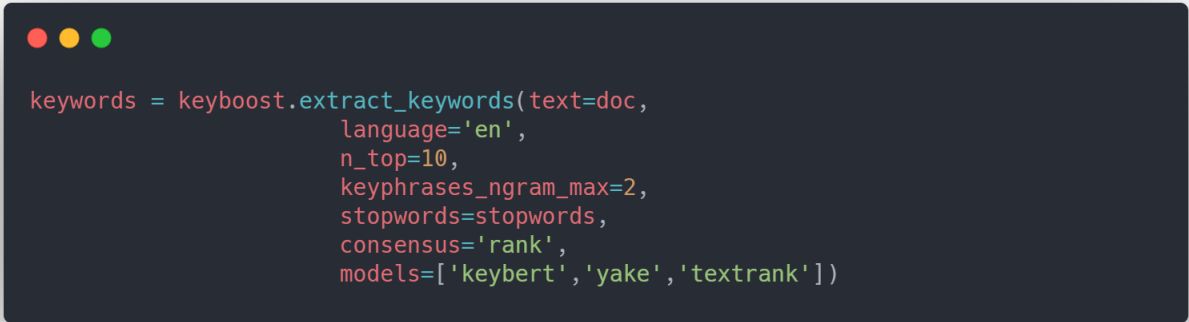
```
selected_models = ['yake', 'textrank']

keywords = keyboost.extract_keywords(text=doc,
                                     language='en',
                                     n_top=10,
                                     keyphrases_ngram_max=2,
                                     stopwords=stopwords,
                                     consensus='statistical',
                                     models=selected_models)
```

FIGURE 5.4 – Modifier les sous-modèles

## 5.5 Type de consensus

*keyBoost* propose deux types de consensus pour consolider les prédictions des différents modèles sous-jacents : *statistical consensus* et *ranking consensus*. Par défaut, il est recommandé d'utiliser le *statistical consensus* car il est susceptible de vous donner les meilleurs résultats la plupart du temps. Cependant, si vous voulez quand même essayer le *ranking consensus*, vous pouvez directement définir l'argument *consensus* sur *rank* :



```
keywords = keyboost.extract_keywords(text=doc,
                                     language='en',
                                     n_top=10,
                                     keyphrases_ngram_max=2,
                                     stopwords=stopwords,
                                     consensus='rank',
                                     models=['keybert', 'yake', 'textrank'])
```

FIGURE 5.5 – Modifier le consensus

### 5.5.1 Informations additionnelles pour le *statistical consensus*

En raison de la nature du *statistical consensus*, il peut parfois revenir sur le *ranking consensus* si les distributions des scores produits par les modèles sous-jacents ne répondent pas à certaines exigences. Vous pouvez vérifier rapidement l'exhaustivité de la procédure de consensus statistique (c'est-à-dire pas de repli sur le consensus de classement) avec la première ligne de code. Enfin, si le *statistical consensus* est complet et que vous souhaitez vérifier les scores respectifs des mots-clés sortis, vous pouvez simplement taper la seconde ligne de code.

```
keyboost.is_statistical_consensus_completed  
  
keyboost.statistical_consensus_scores
```

FIGURE 5.6 – Informations supplémentaires

## 5.6 Modeles d'*embedding*

Les emdeddings font partie de la procédure de consensus et sont élément central pour l'extraction des mots-clés KeyBERT. Bien qu'il soit recommandé d'utiliser les modèles à la pointe que sont *paraphrase-MiniLM-L6-v2* ou *paraphrase-multilingual-MiniLM-L12-v2* pour l'anglais et toutes les autres langues, vous pouvez spécifier n'importe quel modèle de la catégorie *sentence transformers* disponible [ici](#). Il vous suffira simplement de spécifier le nom à l'instanciation de l'objet *KeyBoost*. Par exemple, avec le modèle pré-entraîné *stsb-distilroberta-base-v2* (si c'est la première fois que vous choisissez ce modèle, il se téléchargera automatiquement) :

```
keyboost = KeyBoost('stsb-distilroberta-base-v2')  
  
keywords = keyboost.extract_keywords(text=doc,  
                                     language='en',  
                                     n_top=10,  
                                     keyphrases_ngram_max=2,  
                                     stopwords=stopwords,  
                                     consensus='statistical',  
                                     models=['keybert', 'yake', 'textrank'])
```

FIGURE 5.7 – Modifier le *sentence transformer*

# Bibliographie

- [1] Y. Gallina, F. Boudin, and B. Daille, “Large-Scale Evaluation of Keyphrase Extraction Models,” in *ACM/IEEE Joint Conference on Digital Libraries (JCDL)*, (Wuhan, China), Aug. 2020.
- [2] K. S. Hasan and V. Ng, “Automatic keyphrase extraction : A survey of the state of the art,” in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1 : Long Papers)*, (Baltimore, Maryland), pp. 1262–1273, Association for Computational Linguistics, June 2014.
- [3] R. Campos, V. Mangaravite, A. Pasquali, A. Jorge, C. Nunes, and A. Jatowt, “Yake! keyword extraction from single documents using multiple local features,” *Inf. Sci.*, vol. 509, pp. 257–289, 2020.
- [4] M. Grootendorst, “Keybert : Minimal keyword extraction with bert.,” 2020.
- [5] R. Mihalcea and P. Tarau, “TextRank : Bringing order into texts,” in *Proceedings of EMNLP-04 and the 2004 Conference on Empirical Methods in Natural Language Processing*, July 2004.