MOTS : un outil modulaire pour le résumé automatique

Valentin Nyzam[1] Christophe Rodrigues[2] Aurélien Bossard[1]
(1) Laboratoire d'Informatique Avancée de Saint-Denis, Université Paris 8
2 rue de la Liberté, 93526 Saint-Denis, France
(2) De Vinci Research Center, ESILV
12 avenue Léonard de Vinci, 92400 Courbevoie, France
v.nyzam@iut.univ-paris8.fr, christophe.rodrigues@devinci.fr,
a.bossard@iut.univ-paris8.fr

RÉSUMÉ

Cet article présente un système open source et modulaire pour le résumé automatique : MOTS, développé en Java. Son architecture permet d'implémenter et tester de nouvelles méthodes de résumé automatique et de les comparer avec des méthodes existantes dans un cadre unifié. Ce système, le premier complètement modulaire pour le résumé automatique permet à l'heure actuelle de définir plus de cent combinaisons de modules afin de résumer automatiquement des textes en langage naturel.

ABSTRACT

MOTS : A Modular Framework for Automatic Summarization

This paper presents an open source and modular system for automatic summarization (AS) : MOTS, written in Java. Its architecture allows to implement and test new methods and to ease comparison with already implemented methods in an unified framework. It is the first completely modular system for AS and allows to summarize texts written in natural language using more than a hundred combinations of modules.

1 Introduction

L'évaluation de l'apport de différentes méthodes de résumé automatique, une discipline étudiée depuis les années 1950 (Luhn, 1958), est compliquée. En effet, les techniques de résumé automatique, quand elles sont rendues publiques, sont souvent incluses dans un système plus large qui comprend des pré et post-traitements ainsi que des ressources externes. Ceux-ci influent énormément sur la qualité des résumés produits. Il est donc important d'évaluer les techniques de résumé automatique dans un cadre commun afin de pouvoir les comparer précisément et juger de leur efficacité sur différents types de données.

Dans cet article, nous proposons une solution à ce problème : un outil de résumé automatique complètement modulaire et open source développé en Java qui permet de brancher ou débrancher les pré et post traitements. Cet outil permet également de bénéficier rapidement de *baselines* afin d'accélérer et de faciliter la recherche dans le domaine du résumé automatique.

Un tel outil est une réelle nouveauté. En effet, parmi les systèmes disponibles, on peut distinguer deux catégories : les systèmes issus d'une recherche originale et publiés par leurs auteurs comme MEAD [1]

1. http://www.summarization.com/mead/

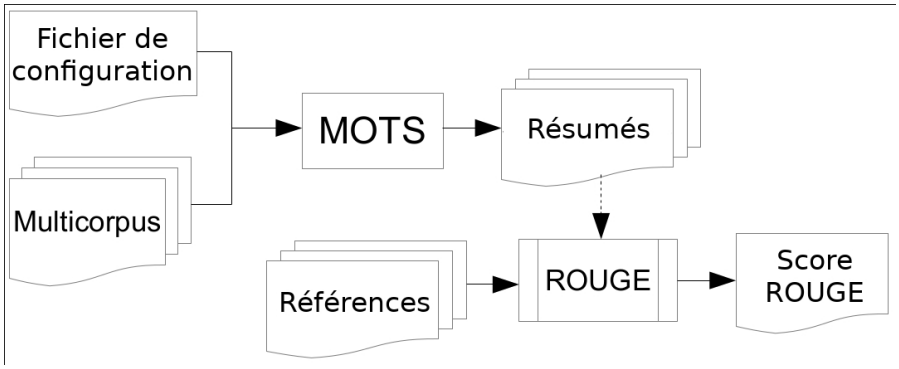

FIGURE 1 – Utilisation du système MOTS

(Radev *et al.*, 2004a), ICSISUMM [2] (Gillick *et al.*, 2009) ou encore MUSEEC [3] (Litvak *et al.*, 2016), qui n'implémentent que la ou les méthodes des auteurs et comprennent des pré et post-traitements spécifiques ; et les systèmes qui ré-implémentent plusieurs méthodes de résumé automatique, comme Sumy [4] et PKUSumSum [5] (Zhang *et al.*, 2016), qui en implémentent respectivement sept et dix. Cependant, les systèmes, y compris dans cette dernière catégorie, ont une modularité limitée au choix de la méthode de résumé automatique.

2 MOTS : Outil modulaire pour le résumé automatique

Pour se démarquer des systèmes cités en Section 1, afin de proposer une vraie solution à l'évaluation de l'apport de différents pré et post-traitements et pour permettre de l'adapter rapidement à une tâche spécifique, notre système de résumé automatique est complètement modulaire sur la totalité de la chaîne de traitement. Chacun de ses composants appartient à une classe spécifique qui code pour un rôle spécifique. Malgré son architecture complexe, MOTS reste facile à utiliser pour l'utilisateur final ; il ne nécessite que deux entrées : un corpus et un fichier de configuration (cf figure 1). MOTS est le plus générique possible afin qu'il puisse gérer ou être adapté à n'importe quel type de résumé automatique : extractif, semi-extractif [6] ou même entièrement abstractif.

2.1 Chaîne de traitement

L'architecture de MOTS est présentée en figure 2. Un multicorpus est ainsi défini comme un ensemble de corpus qui peuvent être résumés indépendemment les uns des autres. Un corpus peut être composé de un ou plusieurs textes, donc MOTS gère les résumés mono et multidocument. Dans cette même figure, un "modèle de résumé" est composé de trois étapes : pré-traitements, méthode de résumé et post-traitements. Une méthode de résumé est, elle, décomposée en quatre étapes qui suivent un schéma classique de résumé extractif ou semi-extractif. Premièrement, la méthode de résumé doit définir comment les "tokens" sont identifiés et indexés par le système. Ensuite, la méthode définit la représentation des phrases (*sentence characteristics*). Puis elle peut évaluer les phrases (toutes

2. https://code.google.com/archive/p/icsisumm
3. https://bitbucket.org/elenach/onr_gui/wiki/Home
4. https://github.com/miso-belica/sumy
5. https://github.com/PKULCWM/PKUSUMSUM/
6. résumé automatique semi-extractif : combinaison d'extraction et de traitements comme la compression ou la paraphrase.

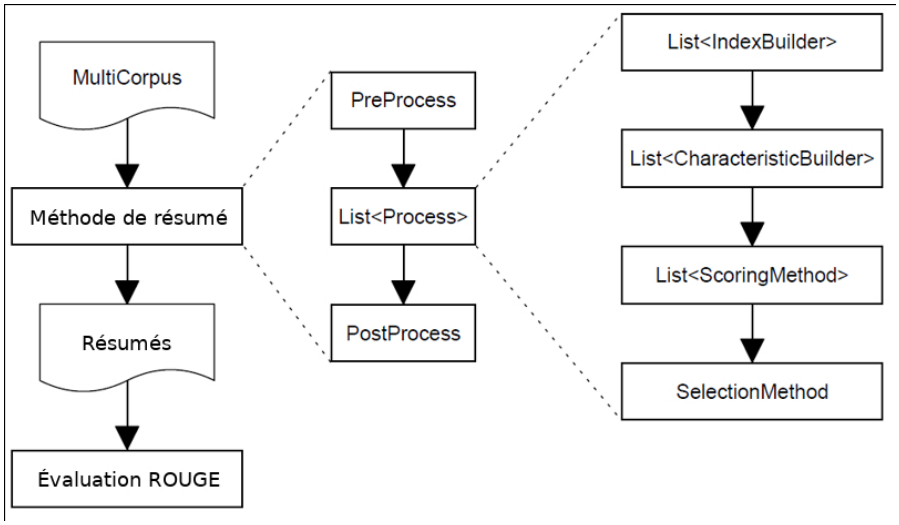

FIGURE 2 – Chaîne de traitement modulaire

les méthodes extractives ne requièrent pas d'évaluer les phrases). C'est l'étape "*sentence scoring*". Finalement, les phrases sont sélectionnées pour apparaître dans le résumé (*selection method*). Une méthode peut avoir de multiples constructeurs d'index, de caractéristiques et de multiples méthodes de *scoring* afin de combiner différentes approches. Les résumés produits par un modèle de résumé peuvent être évalués en utilisant ROUGE (Lin, 2004) qui est encapsulé dans notre système.

2.2 Modularité et implémentation pour MOTS

L'implémentation d'un nouveau modèle de résumé dans MOTS se réalise soit seulement en définissant un nouveau fichier de configuration qui fait appel aux modules existants, soit en définissant au préalable de nouveaux modules. Ces modules, ou traitements atomiques sont indépendants les uns des autres et la communication entre eux est gérée par la classe "Process" qui contrôle leur exécution et compatibilité. Les entrées et sorties des traitements atomiques sont spécifiées par héritage de méthodes et interfaces prédéfinies et décrites dans la documentation de MOTS, qui permettent à la classe "Process" d'utiliser des méthodes pour adapter les entrées et sorties de chaque traitement atomique. Les traitements atomiques sont indépendants les uns des autres et suivent des règles de compatibilité ; cette organisation du code rend MOTS complètement modulaire.

Toute personne désirant implémenter un nouveau traitement pour MOTS peut donc intervenir à n'importe quelle étape de la chaîne de traitement en suivant les règles définies par les interfaces idoines. De nombreux traitements de base utilisés par la majorité des systèmes de résumé automatique ont déjà été implémentés ; l'implémentation de nouvelles méthodes de résumé automatique en est facilitée.

L'outil MOTS est ouvert à tous les contributeurs sous licence GPL.

2.3 Algorithme génétique d'optimisation d'hyperparamètres

Les méthodes de résumé automatique comportent des paramètres qui influent sur la qualité des résumés générés. Selon (Litvak *et al.*, 2010; Bossard & Rodrigues, 2011), un algorithme génétique

```
<CONFIG>
  <TASK ID="1">
    <OPTION NAME="DampingParameter">true</OPTION>
    <OPTION NAME="GraphThreshold">true</OPTION>
    <OPTION NAME="Lambda">false</OPTION>
  </TASK>
</CONFIG>
```

FIGURE 3 – Exemple de fichier de configuration de l'algorithme génétique pour la méthode LexRank : seuls le *daping factor*, lambda et le seuil de similarité sont optimisés.

```
<CONFIG>
  <TASK ID="1">
    <LANGUalgorithme génétiqueE>english</LANGUalgorithme génétiqueE>
    <INPUT_PATH>doc</INPUT_PATH>
    <OUTPUT_PATH>doc/output</OUTPUT_PATH>
    <MULTITHREADING>true</MULTITHREADING>
    <PROCESS>
      <OPTION NAME="CorpusIdToSummarize">all</OPTION>
      <OPTION NAME="ReadStopWords">false</OPTION>
      <INDEX_BUILDER NAME="TF_IDF.TF_IDF">
      </INDEX_BUILDER>
      <CARACTERISTIC_BUILDER NAME="vector.query.Centroid">
        <OPTION NAME="NbMaxWordInCentroid">50</OPTION>
      </CARACTERISTIC_BUILDER>
      <SCORING_METHOD NAME="QuerySimilarity">
        <OPTION NAME="SimilarityMethod">
          JaccardSimilarity
        </OPTION>
      </SCORING_METHOD>
      <SUMMARIZE_METHOD NAME="MMR">
        <OPTION NAME="CharLimitBoolean">true</OPTION>
        <OPTION NAME="Size">100</OPTION>
        <OPTION NAME="Lambda">0.7</OPTION>
        <OPTION NAME="SimilarityMethod">
          JaccardSimilarity
        </OPTION>
      </SUMMARIZE_METHOD>
    </PROCESS>
    <ROUGE_EVALUATION>
      <ROUGE-MEASURE>
        ROUGE-1 ROUGE-2 ROUGE-SU4
      </ROUGE-MEASURE>
      <ROUGE-PATH>
        lib/ROUGE-1.5.5/RELEASE-1.5.5
      </ROUGE-PATH>
      <MODEL-ROOT>models</MODEL-ROOT>
      <PEER-ROOT>systems</PEER-ROOT>
    </ROUGE_EVALUATION>
  </TASK>
</CONFIG>
```

FIGURE 4 – Exemple d'un fichier de configuration pour MOTS

peut améliorer les résultats d'une méthode de résumé automatique donnée afin de construire un système de résumé automatique plus performant ou plus spécialisé. MOTS intègre un algorithme génétique pour optimiser les paramètres des méthodes de résumé automatique. Nous avons défini une syntaxe d'*adn* qui est utilisée pour décrire quels sont les paramètres d'une méthode à optimiser. Ces paramètres, définis dans les traitements atomiques, peuvent être ou non optimisés par l'algorithme génétique selon un fichier de configuration en xml passé en entrée de MOTS dont un exemple est donné en figure 3.

3 Méthodes implémentées

3.1 Indexation

MOTS inclut des méthodes pour indexer selon des unigrammes ou n-grammes, selon leur fréquence ou leur *tf.idf* (Salton & Buckley, 1988).

Est également implémentée une méthode d'indexation fondée sur *LDA* (Blei *et al.*, 2003) : k *topics* sont identifiés par une phase d'analyse Dirichlet latente sur les documents à résumer. Chaque *token* se voit alors attribuer une probabilité de distribution sur ces k *topics*.

Query LDA étend *LDA* en construisant la distribution des topics pour les documents. Les documents peuvent alors servir de requête pour évaluer la pertinence d'un fragment textuel (plus un fragment aura une distribution de *topics* proche de celle des documents, plus il sera pertinent).

Word embeddings génère les plongements lexicaux pour chaque *token* selon la méthode de (Mikolov *et al.*, 2013). Les *tokens* sont alors représentés par un vecteur de nombres décimaux qui exprime leur sémantique.

3.2 Caractérisation

L'étape de caractérisation sert à obtenir une représentation de séquences de *tokens* à partir de la représentation des *tokens* eux-mêmes.

MOTS implémente des méthodes vectorielles pour les phrases, les documents et les corpus : sac de mots *tf.idf*, construction d'un *centroïde* comme décrit dans (Radev *et al.*, 2004b), ainsi que leur extension matricielle. MOTS implémente également des caractéristiques fondées sur les graphes : graphes de co-occurrence (Rousseau & Vazirgiannis, 2013), K-core (Batagelj & Zaversnik, 2003), ainsi que des méthodes de regroupement automatique de phrases fondées sur les caractéristiques obtenues par une autre des méthodes de caractérisation définies.

3.3 Notation de fragments textuels

La notation de fragments textuels utilise les représentations calculées au préalable par l'étape de caractérisation afin de noter les fragments en vue de leur éventuelle sélection dans le résumé.

Les méthodes de notation suivantes sont implémentées dans MOTS :

- *LexRank* (Erkan & Radev, 2004) se fonde sur la notion de popularité dans un graphe afin de détecter les fragments les plus centraux d'un ou plusieurs documents (Erkan & Radev, 2004) ;
- *tf.idf threshold* somme les *tf.idf* des tokens d'un fragment au dessus d'un seuil ;
- *QuerySimilarity* évalue un fragment de texte vis-à-vis d'une requête vectorielle construite dans l'étape précédente – cela permet par exemple de réaliser la méthode centroïde (Radev *et al.*, 2004b) dans MOTS.

3.4 Sélection

Méthodes incrémentales MOTS implémente trois méthodes de sélection incrémentale de phrases.

BestIsBetter extrait naïvement les meilleurs fragments selon une des notations définies en §3.3 et risque donc, surtout en contexte multidocument, de produire des résumés redondants.

MMR sélectionne à chaque itération la phrase qui est à la fois la plus centrale et la moins similaire aux phrases déjà sélectionnées selon la méthode de (Carbonell & Goldstein, 1998), donc le fragment qui maximise la formule suivante :

$$MMR(D,R) = \underset{F_i \in D \setminus R}{\arg\max}[\lambda \times centralite(F_i) - (1-\lambda) \max_{F_j \in R}(sim(F_i, F_j))]$$

où D est l'ensemble des documents à résumer et R l'ensemble des fragments déjà extraits dans le résumé, centralité une mesure de centralité et sim une mesure de similarité entre fragments.

CSIS extrait itérativement le meilleur fragment dont la similarité à un fragment déjà sélectionné n'excède pas un seuil prédéfini, comme décrit dans (Radev *et al.*, 2004a).

Méthodes d'exploration l'outil inclut également des méthodes d'exploration de l'espace des résumés candidats. Ces méthodes permettent de pallier le principal défaut des méthodes incrémentales : les résumés produits sont trop dépendants de la première phrase sélectionnée.

ILP optimise une fonction de score linéaire sous contraintes en nombres entiers (Gillick & Favre, 2009). La fonction à optimiser est une simple somme pondérée des *tokens* avec leur nombre d'occurrences comme poids. La méthode d'indexation définit donc quel *token* sera pris en compte dans la fonction objectif d'*ILP*. Cette méthode, avec la fonction objectif fondée sur les bigrammes, est reconnue à la fois pour sa rapidité d'exécution et son efficacité sur des tâches de résumé multidocument.

Knapsack utilise un algorithme dynamique de résolution du problème du sac à dos décrit dans (McDonald, 2007) afin d'optimiser une fonction objectif en prenant en compte la contrainte de la taille des résumés.

Genetic implémente la méthode décrite dans (Bossard & Rodrigues, 2017), qui optimise une fonction objectif grâce à un algorithme évolutionnaire. Cet algorithme évolutionnaire a l'avantage de optimiser n'importe quelle fonction objectif a un temps d'exécution conséquent.

Pour ces deux dernières méthodes, différentes fonctions objectif sont disponibles : similarité au document, à une requête, divergence distributionnelle... Tout comme pour l'architecture globale de MOTS et l'interface de notation de fragments textuels, il existe une interface générique dont doivent hériter toute nouvelle fonction objectif.

Sont également incluses des méthodes de résumé automatique par l'apprentissage par renforcement, qui doit explorer un espace de solution conséquent et montre les limites de ce type de méthodes appliquées au résumé automatique (Sutton & Barto, 1998; Ryang & Abekawa, 2012).

Méthodes par abstraction Les méthodes de résumé par abstraction connaissent un regain d'intérêt depuis 2015 (Rush *et al.*, 2015). Ce regain d'intérêt est principalement dû aux capacités génératives des réseaux de neurones récurrents, qui ont montré une efficacité certaine sur la traduction automatique. Ces dernières années ont vu beaucoup d'évolution dans les modèles d'apprentissage afin de les adapter au mieux au résumé automatique malgré la difficulté principale inhérente à ce domaine : un espace de recherche très étendu par rapport à la traduction automatique (il s'agit pour résumer de "traduire" un ou plusieurs documents en plusieurs phrases, et non une phrase en une autre).

Étant donné le renouveau des méthodes abstractives pour le résumé automatique, nous ne pouvions pas uniquement dédier MOTS au résumé extractif et semi-extractif. Nous y avons donc également implémenté un module qui permet d'appeler un système de résumé par abstraction et d'intégrer ses résultats à MOTS. Cette intégration n'est pas évidente : les systèmes par abstraction, notamment ceux à base de réseaux de neurones récurrents dont le modèle a été appris au préalable peuvent produire des mots en dehors du vocabulaire des documents d'origine. Ces mots doivent alors être intégrés a posteriori à l'index afin de pouvoir réaliser les post-traitements dans MOTS sur les résumés produits par ce module.

4 Utiliser MOTS

Le système MOTS est conçu pour être facile à utiliser et à y contribuer malgré son architecture relativement complexe. En tant qu'utilisateur final, seules sont nécessaires la mise au format du

corpus ainsi que l'écriture d'un fichier de configuration qui décrit le modèle de résumé qui servira à générer les résumés. Pour plus de simplicité, des fichiers de configuration sont prédéfinis dans le système. La figure 4 présente un fichier de configuration qui décrit un modèle qui utilise le score des phrases par la méthode "centroïde" et leur sélection par la méthode "MMR".

Ainsi, avec MOTS, l'installation d'un système unique et une mise au format du corpus suffisent pour récupérer les résumés produits par plusieurs méthodes état de l'art, contre l'installation de plusieurs systèmes et plusieurs conversions du corpus pour le même résultat sans MOTS. Les résultats différeront forcément légèrement des systèmes d'origine. En effet, les pré-traitements utilisés ne sont pas les mêmes, et les systèmes d'origine peuvent différer légèrement dans l'utilisation de procédures annexes non décrites dans les articles. Cependant, implémenter une nouvelle méthode de résumé au sein de MOTS permet de la comparer à des méthodes existantes dans exactement les mêmes conditions.

5 Évaluation

5.1 Protocole

Nous décrivons ici le protocole d'évaluation de MOTS, qui permet d'évaluer les méthodes implémentées et de les comparer quand c'est possible aux systèmes d'origine qu'elles émulent. Les méthodes sont évaluées sur les corpus DUC 2006/2007, TAC 2008/2009/2010 sur leur partie standard et sans prise en compte des requêtes et dont les caractéristiques sont données dans le tableau 4.

5.2 Méthodes évaluées

Toutes les méthodes partagent les mêmes pré et post-traitements : StanfordNLP pour le découpage en mots/phrases (Manning *et al.*, 2014) et le *stemmer* de Porter pour la racinisation. Le tableau 1 détaille les processus atomiques utilisés dans les méthodes évaluées à des fins de reproductibilité.

JSBigram Knapsack utilise la méthode d'exploration *Knapsack* décrite dans (McDonald, 2007) pour optimiser une métrique d'évaluation de résumé automatique fondée sur la distance de Jensen-Shannon décrite dans (Louis & Nenkova, 2009).

JSBigram Genetic utilise la méthode d'optimisation évolutionnaire *Genetic* pour optimiser la même métrique que "JSBigram Knapsack" et émule donc la méthode décrite dans (Bossard & Rodrigues, 2017).

JSBigram Reinforcement utilise la méthode d'apprentissage par renforcement décrite dans (Ryang & Abekawa, 2012) pour optimiser la même métrique que les deux méthodes précédentes.

Bigram ILP émule la méthode de résumé automatique par programmation linéaire en nombres entiers de (Gillick & Favre, 2009).

LexRank MMR utilise la méthode de résumé automatique LexRank (Radev *et al.*, 2004a) comme score de phrases et MMR (Carbonell & Goldstein, 1998) pour les extraire dans le résumé.

LexRank MMR opt optimise sur le corpus TAC 2009 les trois paramètres *damping factor, epsilon et seuil de similarité* de la méthode "LexRank MMR" grâce à l'algorithme génétique de MOTS.

	Indexation	Caractéristiques	Score	Sélection
JSBigram Knapsack	Word, 2-gram	-	-	Knapsack (JS metric)
JSBigram Genetic	Word, 2-gram	-	-	Genetic (JS metric)
Bigram ILP	Word, 2-gram	-	-	ILP
LexRank MMR	Word	tf.idf vector	LexRank (Jaccard)	MMR (cosine)
TfIdf MMR Cosine	Word	tf.idf document	query sim (cosine)	MMR (cosine)
LDA	Query LDA	mean vector sent	query sim (JS)	BestIsBetter
Bigram Centr. MMR	Word, 2-gram	centroid	query sim (Jaccard)	MMR (Jaccard)
KCore Query	Word	KCore query	query sim (Jaccard)	MMR (Jaccard)
JSBigram Reinforc.	Word, 2-gram	tf.idf vector	-	Reinforc.Learn. (JS)
W2V LexRank MMR	Word embeddings	mean vector sent	LexRank (cosine)	MMR (cosine)

TABLE 1 – Résumé des processus atomiques utilisés dans les méthodes évaluées

TF-IDF MMR Cosinus émule la méthode du centroïde (Radev *et al.*, 2004b) pour évaluer les phrases, et MMR pour les sélectionner.

LDA utilise l'analyse Dirichlet latente pour représenter les phrases et le corpus comme des distributions de topics (Blei *et al.*, 2003). La distribution du document est utilisée comme requête.

Bigram Centroid MMR est la même méthode que centroïde (Radev *et al.*, 2004b) mais utilise les bigrammes et non les unigrammes comme tokens.

KCore Query consiste à construire un graphe de co-occurrences des tokens comme dans (Rousseau & Vazirgiannis, 2013). La dégénérescence du graphe est calculée pour obtenir une décomposition en K-core (Batagelj & Zaversnik, 2003). Le meilleur K-core est utilisé comme requête et MMR comme méthode d'extraction.

W2V LexRank MMR utilise le vecteur moyen des plongements lexicaux d'une phrase comme caractéristique, LexRank/MMR pour extraire les phrases.

5.3 Baselines

Nous incluons les résultats obtenus par quatre systèmes externes publiés par leurs auteurs : **ICSISUMM query/ICSISUMM wo query** [7] décrit dans Gillick *et al.* (2009) et comparable à la méthode "Bigram ILP". "ICSISUMM wo query" débranche un filtre dans ICSISUMM lié à la proximité avec les requêtes des corpus DUC/TAC non utilisées dans notre système.

Genetic official : le résumé automatique par algorithme évolutionnaire (Bossard & Rodrigues, 2017) comparable à notre méthode "JSBigram Genetic".

MEAD [8] avec l'algorithme "centroïde", comparable à notre méthode "Centroid MMR".

5.4 Résultats

L'évaluation réalisée avec l'outil ROUGE et les paramètres que Graham (2015) a trouvé les plus corrélés aux scores manuels [9] est présentée en tableau 2. Ces résultats semblent cohérents avec les

7. https://github.com/benob/icsisumm
8. http://www.summarization.com/mead/
9. Les paramètres exacts sont : -n 2 -x -m -c 95 -r 1000 -f A -p 1 -t 0 -a -s

Corpora	D2006	D2007	T2008	T2009	T2010
ICSISUMM query	.07617	**.09952**	**.10625**	**.10108**	.08343
ICSISUMM wo query	.07286	.09753	.10286	.09150	.08973
Genetic official	**.07678**	.08591	.10448	.09537	**.09892**
MEAD	.05149	.06323	.05075	.04561	.06427
JSBigram Knapsack	.06882	**.09015**	**.10923**	.08732	.08853
JSBigram Genetic	**.07357**	.08399	.10544	.08576	.08900
Bigram Centr. MMR	.07328	.08276	.10658	.08112	.08848
Bigram ILP	.06977	.08077	.10123	.08251	**.09152**
LexRank MMR opt	.06952	.07805	.09150	**.09369**	.07714
LexRank MMR	.07094	.07917	.09370	.08609	.08128
JSBigram Reinforc.	.06779	.07160	.09637	.07310	.07328
KCore Query MMR	.05908	.07020	.08659	.06665	.07721
TFIDF MMR Cosine	.05786	.07291	.07990	.06383	.07110
LDA	.05659	.07113	.07471	.07275	.06919
W2V LexRank MMR	.06504	.06723	.04501	.04564	.05201

TABLE 2 – Scores ROUGE-2 sur tous les corpus

Corpora	D2006	D2007	T2008	T2009	T2010
ICSISUMM	651s	337s	95s	85s	56s
Genetic official	7121s	5123s	2047s	1951s	1911s
MEAD	186s	86s	74s	85s	56s
JSBigram Knapsack	1941s	1076s	281s	250s	217s
JSBigram Genetic	7950s	5850s	2492s	2375s	2251s
Bigram ILP	338s	181s	48s	42s	60s
Bigram Centr. Jacc MMR	74s	33s	12s	11s	10s
LexRank MMR Jacc	37s	20s	7s	8s	7s
KCore Query MMR Jacc	21s	12s	6s	6s	7s
JSBigram Reinforc.	1041s	672s	286s	266s	263s
TFIDF MMR Cosine	18s	12s	5s	6s	6s
LDA	58s	41s	21s	21s	20s
W2V LexRank MMR	134s	131s	104s	101s	103s

TABLE 3 – Temps d'exécution sur tous les corpus

méthodes état de l'art testées, même si de légères différences dans les résultats semblent valider l'importance des pré et post-traitements dans la qualité des résumés produits.

La méthode de résumé automatique fondée sur les plongements lexicaux n'est pas optimale et est peu performante. La comparaison de "ICSISUMM query" et "ICSISUMM wo query" montre l'importance de pré-traitements adaptés à la tâche : "ICSISUMM query" prend en compte les requêtes de chaque *topic* tandis que la prise en compte des requêtes est désactivée dans "ICSISUMM wo query". Nos meilleurs modèles de résumé automatique sont en effet meilleurs en moyenne que "ICSISUMM wo query" mais sont dépassés par "ICSISUMM query".

L'algorithme génétique embarqué par MOTS a été utilisé pour optimiser la méthode 'LexRank MMR opt" sur TAC2009. Les scores sur ce corpus sont nettement améliorés bien que l'amélioration soit limitée à ce corpus. Les hyperparamètres que nous avons optimisés sont uniquement ceux de l'algorithme de sélection, et ne sont sans doute pas assez généralisables. Pour obtenir une optimisation par l'algorithme génétique plus convaincante, il serait sans doute préférable d'inclure des hyperparamètres

liés au traitement des données en entrée, comme des poids différents pour chaque catégorie de *tokens* (entités nommées, verbes, adverbes...).

	DUC2006	DUC2007	TAC2008	TAC2009	TAC2010
topics	50	45	48	44	46
words/topic	17728	13693	6234	6679	5790

TABLE 4 – Nombre de *topics* et de mots par *topic* dans chaque corpus

Le Tableau 3 montre le temps d'exécution de chaque modèle sur chaque corpus. Les tâches DUC et TAC diffèrent das le nombre de mots à produire par résumé et dans le nombre de mots en entrée (cf tableau 4). Les comparer donne donc une idée de possibilités de passage à l'échelle des modèles de résumé automatique.

Notre évaluation a mis en valeur une nouvelle méthode qui dépasse les systèmes fondées sur les algorithme génétique et ILP : "JSBigram Knapsack". Cela valide l'intérêt de la modularité de notre système, qui a permis de tester facilement différentes combinaisons de modules, dont celle-ci.

5.5 Discussion

Les résultats mettent en évidence des différences de scores assez importantes, notamment en ce qui concerne les systèmes "MEAD" et "Bigram Centroid MMR". Ce dernier se révèle d'une fois et demie à deux fois plus performant en termes de scores ROUGE que MEAD. MEAD par défaut une combinaison de scores, dont "Centroïde" et la position des phrases dans le document. Ce dernier score est sensé améliorer les résultats de "Centroïde" seul, il est donc d'autant plus étonnant de constater de telles différences de score ROUGE.

Nous ne pouvons les expliquer que par les pré-traitements des deux systèmes. MOTS utilise par défaut StanfordNLP pour le découpage en mots et en phrases ainsi que le *stemmer* de Porter pour raciniser les *tokens*, tandis que "MEAD" utilise les mots pleins. Ces différences de score ne font qu'appuyer notre hypothèse initiale, à savoir que les pré et post-traitements ont une influence importante sur les résultats d'un système de résumé automatique.

6 Conclusion et perspectives

Cet article présente MOTS, le premier système complètement modulaire pour le résumé automatique. Ce système est *open source* et librement disponible. Il vise à faciliter la comparaison dans un cadre unifié de nouvelles méthodes de résumé automatique avec des méthodes existantes et implémentées dans MOTS. MOTS est disponible sur GitHub [10], pour que chacun puisse y contribuer. MOTS contient un algorithme génétique qui optimise les hyperparamètres des modèles de résumé automatique.

La comparaison des résultats des modèles de résumé automatique de MOTS avec les résultats des systèmes d'origine qu'ils émulent montrent le besoin d'évaluations différentes qui permettent de quantifier l'apport réel de méthodes de résumé en les isolant des pré et post-traitements.

10. Nom sur github changé pour préserver l'anonymat

Même si nous n'avons évalué que dix méthodes de résumé automatique, la modularité de MOTS permet, en combinant les processus atomiques, de définir plus d'une centaine de méthodes de résumé automatique. Nous avons également présenté une méthode de résumé automatique inédite fondée sur la décomposition en K-Cores avec des premiers résultats mitigés. La modularité de MOTS nous a permis de tester aisément différentes combinaisons de processus atomiques et de mettre en valeur une nouvelle méthode de résumé automatique, rapide et efficace, qui utilise un algorithme de programmation dynamique de résolution du problème du sac à dos guidé par la divergence de Jensen-Shannon.

Nous avons implémenté un module qui permet d'appeler un système de résumé abstractif externe, récupérer ses résultats et les intégrer à MOTS pour d'éventuels post-traitements. Ce module permet ainsi d'utiliser les avancées récentes dans le domaine des réseaux de neurones profonds et leur application au résumé automatique (Hua & Wang, 2017; Tan *et al.*, 2017; Zhou *et al.*, 2017; Chopra *et al.*, 2016).

Le système MOTS est ouvert à tous les contributeurs. Nous implémentons actuellement d'autres méthodes état de l'art (p.e. les fonctions submodulaires) et améliorons certaines caractéristiques (p.e. les plongements lexicaux). Nous espérons que la communauté du TAL trouvera MOTS utile et qu'il gagnera de nouveaux contributeurs.

Remerciements

Ce travail a bénéficié d'une aide de l'Agence Nationale de la Recherche portant la référence ANR-16-CE38-0008 (projet ANR JCJC ASADERA).

Références

BATAGELJ V. & ZAVERSNIK M. (2003). An o(m) algorithm for cores decomposition of networks. *CoRR*, **cs.DS/0310049**.

BLEI D. M., NG A. Y. & JORDAN M. I. (2003). Latent dirichlet allocation. *Journal of machine Learning research*, **3**(Jan), 993–1022.

BOSSARD A. & RODRIGUES C. (2011). Combining a multi-document update summarization system–cbseas–with a genetic algorithm. In *Combinations of intelligent methods and applications*, p. 71–87. Springer.

BOSSARD A. & RODRIGUES C. (2017). An evolutionary algorithm for automatic summarization. In *Proceedings of the International Conference Recent Advances in Natural Language Processing, RANLP 2017*, p. 111–120, Varna, Bulgaria : INCOMA Ltd.

CARBONELL J. & GOLDSTEIN J. (1998). The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, p. 335–336 : ACM.

CHOPRA S., AULI M. & RUSH A. M. (2016). Abstractive sentence summarization with attentive recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies*, p. 93–98, San Diego, California : Association for Computational Linguistics.

ERKAN G. & RADEV D. R. (2004). Lexrank : Graph-based lexical centrality as salience in text summarization. *Journal of AIR*, **22**, 457–479.

GILLICK D. & FAVRE B. (2009). A scalable global model for summarization. In *Proceedings of the Workshop on Integer Linear Programming for Natural Langauge Processing*, p. 10–18 : Association for Computational Linguistics.

GILLICK D., FAVRE B., HAKKANI-TÜR D., BOHNET B., LIU Y. & XIE S. (2009). The icsi/utd summarization system at tac 2009. In *Proc. of the Text Analysis Conference workshop, Gaithersburg, MD (USA)*.

GRAHAM Y. (2015). Re-evaluating automatic summarization with bleu and 192 shades of rouge. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, p. 128–137, Lisbon, Portugal : Association for Computational Linguistics.

HUA X. & WANG L. (2017). A pilot study of domain adaptation effect for neural abstractive summarization. In *Proceedings of the EMNLP Workshop on New Frontiers in Summarization*, Copenhagen, Denmark : Association for Computational Linguistics.

LIN C.-Y. (2004). Rouge : A package for automatic evaluation of summaries. In *Text summarization branches out : Proceedings of the ACL-04 workshop*, volume 8 : Barcelona, Spain.

LITVAK M., LAST M. & FRIEDMAN M. (2010). A new approach to improving multilingual summarization using a genetic algorithm. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, p. 927–936 : Association for Computational Linguistics.

LITVAK M., VANETIK N., LAST M. & CHURKIN E. (2016). Museec : A multilingual text summarization tool. In *Proceedings of ACL-2016 System Demonstrations*, p. 73–78 : Association for Computational Linguistics.

LOUIS A. & NENKOVA A. (2009). Automatically evaluating content selection in summarization without human models. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing : Volume 1-Volume 1*, p. 306–314 : Association for Computational Linguistics.

LUHN H. P. (1958). The automatic creation of literature abstracts. *IBM J. Res. Dev.*, **2**(2), 159–165.

MANNING C. D., SURDEANU M., BAUER J., FINKEL J., BETHARD S. J. & MCCLOSKY D. (2014). The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, p. 55–60.

MCDONALD R. (2007). A study of global inference algorithms in multi-document summarization. *Advances in Information Retrieval*, p. 557–564.

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*, p. 3111–3119.

RADEV D. R., ALLISON T., BLAIR-GOLDENSOHN S., BLITZER J., CELEBI A., DIMITROV S., DRABEK E., HAKIM A., LAM W., LIU D. *et al.* (2004a). Mead-a platform for multidocument multilingual text summarization. In *LREC*.

RADEV D. R., JING H., STYŚ M. & TAM D. (2004b). Centroid-based summarization of multiple documents. *Information Processing & Management*, **40**, 919–938.

ROUSSEAU F. & VAZIRGIANNIS M. (2013). Graph-of-word and tw-idf : new approach to ad hoc ir. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, p. 59–68 : ACM.

RUSH A. M., CHOPRA S. & WESTON J. (2015). A neural attention model for abstractive sentence summarization. In *EMNLP*, p. 379–389.

RYANG S. & ABEKAWA T. (2012). Framework of automatic text summarization using reinforcement learning. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, p. 256–265 : Association for Computational Linguistics.

SALTON G. & BUCKLEY C. (1988). Term-weighting approaches in automatic text retrieval. *Information processing & management*, **24**(5), 513–523.

SUTTON R. S. & BARTO A. G. (1998). *Reinforcement learning : An introduction*, volume 1. MIT press Cambridge.

TAN J., WAN X. & XIAO J. (2017). Abstractive document summarization with a graph-based attentional neural model. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1 : Long Papers)*, p. 1171–1181 : Association for Computational Linguistics.

ZHANG J., WANG T. & WAN X. (2016). PKUSUMSUM : A java platform for multilingual document summarization. In *COLING (Demos)*, p. 287–291 : ACL.

ZHOU Q., YANG N., WEI F. & ZHOU M. (2017). Selective encoding for abstractive sentence summarization. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1 : Long Papers)*, p. 1095–1104 : Association for Computational Linguistics.