Université de Montréal
Département de génie informatique et génie logiciel

IFT6010
Introduction au traitement automatique des langues naturelles

Rapport

Soumis par
Ludovic Font,
Lara Haidar-Ahmad

Dépôt git : https://github.com/Flutifioc/ift6010

24 Avril 2015

Contexte

Avec l'accumulation des textes numériques, l'automatisation du traitement de ces textes est une problématique majeure. L'un des grands domaines de ce traitement automatique est l'obtention de résumés. Pour cela, il existe actuellement deux approches : l'approche par extraction, où la machine repère les phrases ou éléments de phrases les plus importants du texte et les recombine pour former le résumé, et l'approche par abstraction, où le processus passe par une étape de "compréhension" du texte, à partir de laquelle de nouvelles phrases sont générées.

L'approche par abstraction est prometteuse et fournit de meilleurs résultats en général, mais est beaucoup plus difficile à implémenter. Dans ce projet, notre objectif est de mettre en avant les différences fondamentales entre les deux approches. Pour cela, nous allons comparer les résultats obtenus par deux méthodes, une par abstraction (*Opinosis*, par Kavita Ganesan, ChengXiang Zhai and Jiawei Han) et une par extraction (*LexRank*, par Gunes Erkan et Dragomir R. Radev), sur différents types de textes. Nous tenterons ensuite de combiner les deux méthodes afin d'améliorer les résultats.

Méthodologie

1. Environnement et language

Le traitement du texte s'effectuant informatiquement par du travail sur des chaînes de caractères, nous avons naturellement choisi de travailler en Java. Ce langage offre en effet plusieurs outils efficaces pour le traitement des chaînes de caractère. De plus, nous souhaitions pouvoir travailler sur diverses machines pour plusieurs raisons, et la portabilité était donc également importante. Enfin, il s'agit d'un langage que nous connaissons et maîtrisons tous les deux.

Nous avons initialement choisi de travailler sur Eclipse, mais nous avons eu plusieurs problèmes avec cet IDE, et sommes donc passés sur Netbeans. Nos machines

personnelles fonctionnent sous Windows, mais nous avons été amenés par moments à travailler sur les distributions Unix de Poly.

2. Objectifs initiaux

Au départ, nous comptions effectuer le travail suivant :

- Implémenter les deux méthodes (LexRank et Opinosis)
- Interfacer la métrique ROUGE afin de mesurer nos résultats
- Comparer les deux méthodes

Peu après le début du projet, nous avons également eu l'idée suivante : un problème important des méthodes par extraction est qu'elles génèrent souvent du texte redondant. Opinosis, d'un autre côté, est très efficace pour combiner des phrases proches dans leur structure. Nous avons donc ajouté l'étape suivante :

- Utilisation d'Opinosis sur le résumé généré par LexRank et comparaison avec les méthodes individuelles

Nous nous sommes également posés rapidement la question du corpus de test. En effet, ROUGE nécessite, pour juger le résumé d'un texte, un résumé de référence, écrit de préférence par des humains. Nous avons donc demandé l'accès à plusieurs corpus, dont celui du CSTBank (http://clair.si.umich.edu/clair/CSTBank/), qui nous ont permis de faire nos tests.

3. Répartition du travail

Nous avons opté pour une répartition dynamique du travail, en fonction de l'avancement de chacun. Nous avons donc commencé par définir ensemble certaines conventions, par exemple sous quelle forme nos programmes devaient traiter les textes et écrire les résumés, et nous avons chacun commencé à travailler sur une méthode, Lara sur le résumé par extraction, et Ludovic sur le résumé par abstraction.

Lorsque nous avons constaté que la méthode par abstraction serait probablement plus rapide à implémenter, notamment parce que Lara avait trois différentes approches à mettre en place (plus de détails dans la section “Résumé par extraction”), Ludovic a été chargé de mettre en place l’interface entre les deux méthodes : lecture des documents et implémentation de ROUGE, notamment.
Il y a également certains points sur lesquels nous avons travaillé ensemble : discussions initiales, recherche de corpus de test, analyse du résultat et rédaction du rapport.

Résumé par abstraction

Pour commencer, une remarque préliminaire décevante : Opinosis, la méthode que nous avons choisie, n’est pas à proprement parler une méthode par abstraction. Il s’agit d’une méthode par extraction utilisant des éléments d’abstraction. Nous avons tout de même choisi cette méthode pour plusieurs raisons :

- L’approche est originale et amusante
- Les méthodes par abstraction pure sont, en plus d’être rares, soit restreintes à un domaine précis, soit demandent un travail humain non négligeable
- Il s’agit de la première approche que nous avons trouvée, et nous étions déjà bien engagés dans le projet lorsque nous nous sommes aperçus de ce petit souci
- A posteriori, les résultats restent intéressants !

Pour simplifier la lecture, nous nous concentrerons ici sur notre implémentation de la méthode et non sur les détails de son fonctionnement.
La première chose à savoir est que le code d’Opinosis n’est pas disponible. Nous avons donc du envoyer un mail à son inventeur pour y demander accès, et même ainsi, nous n’avons récupéré qu’une clé nous permettant d’envoyer des requêtes sur un serveur distant. Ainsi, notre travail sur cette méthode consiste essentiellement à découper le

texte en phrases, l'annoter, construire et envoyer une requête, et décortiquer les résultats de la requête. Cela nous a d'ailleurs posé quelques soucis, car le pare-feu de Poly bloquait ces requêtes, nous obligeant à travailler chez nous.
Quelques mots concernant l'annotation : Opinosis demande en entrée des phrases où les mots sont syntaxiquement annotés. Pour cela, nous avons choisi de faire appel au tagger POS Stanford, car il s'agit de celui mentionné par l'article.
En soi, la difficulté était donc de comprendre le fonctionnement et le format des requêtes à envoyer au serveur, et réussir à construire des requêtes correctes.
Malheureusement, lorsque nous avons essayé sur de vrais textes, les résultats étaient très décevants. Sur une suite de 70 avis d'utilisateurs sur un même produit, nous n'obtenons que 6 phrases, et, contrairement à ce que promet l'article, elles ne sont pas "réunies" correctement : "the software is easy" et "software is great" auraient du être réunies en "software is easy and great"...

Résumé par extraction

Comme son nom l'indique, cette méthode procède par la construction d'un résumé par l'extraction des phrases les plus pertinentes d'un texte. Pour ce faire, il existe plusieurs approches qui permettent d'attribuer des scores aux phrases d'un texte afin de détecter et sélectionner les phrases les plus importantes.
La méthode de génération de résumé par extraction est une méthode qui fonctionne le mieux sur des textes ou articles, contrairement à la méthode par abstraction qui fonctionne mieux sur des phrases très similaires (comme des commentaires ou des opinions).
Nous nous sommes basés sur un article qui mettait en valeurs essentiellement trois méthodes d'extraction; soit, la méthode des centroïde, la méthode des centralités, ainsi que la méthode *Lexrank*. Les trois algorithmes ont été implémentés dans la classe

ExtractionSummarizer, cependant, c'est la méthode *Lexrank* qui sera utilisé pour nos tests, puisque c'est celle-ci qui génère les résultats les plus prometteurs.

Afin de calculer les scores des phrases, *Lexrank* se base sur la similarité entre les phrases, tout en donnant un poids à chacune d'elles.

Ainsi, pour chaque paire de phrases *(x, y)*, on calcul le *idf-modified-cosine* qui se définit par:

$$\text{idf-modified-cosine}(x, y) = \frac{\sum_{w \in x,y} \text{tf}_{w,x}\text{tf}_{w,y}(\text{idf}_w)^2}{\sqrt{\sum_{x_i \in x}(\text{tf}_{x_i,x}\text{idf}_{x_i})^2} \times \sqrt{\sum_{y_i \in y}(\text{tf}_{y_i,y}\text{idf}_{y_i})^2}}$$

,

où *tfw,x* est la fréquence du mot *w* dans la phrase *x*, et *idfw* le score *idf* attribue au mot w (dont on parlera plus loin dans cette partie).

Cette valeur est un indicateur de la distance entre les deux phrases x et y.

On peut par la suite construire un graphe dans lequel les phrases du texte sont les noeuds et les liens entre les noeuds sont la présence de similarité entre ces phrases; si la valeur du *idf-modified-cosine* d'une paire de phrases est supérieure à un certain seuil, on suppose qu'elles sont similaires et donc qu'elles sont reliées dans le graphe, sinon elles ne sont pas similaires, et ne sont pas directement liées dans le graphe. On peut alors, donner un poids à chacune des phrases, et calculer le score *p* de chacune des phrases par :

$$p(u) = \sum_{v \in adj[u]} \frac{p(v)}{deg(v)}$$

,

où *p(u)* est le score du noeud *u*, *v* les noeuds voisins du noeud *u*, et *deg(v)* le degré du noeud *v* (soit nombre de voisins de ce noeud).

$$\mathbf{p} = \mathbf{B}^{\mathrm{T}}\mathbf{p}$$

Sous forme matricielle, on obtient : , où p est le vecteur des scores des phrases, et *B* la matrice d'adjacence dont les colonnes sont divisées par les degrés de chacun des noeuds .

Ainsi, une fois que la matrice d'adjacence est obtenue, on peut facilement calculer B^T, et calculer le vecteur de distribution stationnaire de Markov afin de trouver p (voir la fonction *powerMetthod* de la classe *ExtractionSummarizer* pour plus d'information).

D'autre part, afin de calculer le *inverse document frequency (idf)* des mots, définit en tant que indicateur de la fréquence des mots dans la langue, nous avons eu recours à un fichier dump *Wikipédia* (de 205593 documents) que nous avons parcourue afin de calculer les scores des mots. Le code du calcul de ce score se trouve dans la classe *WordAnnoter*.

Comparaison

La méthode d'extraction génère des résultats sous forme de diverses phrases similaires qui forment un texte, alors que la méthode d'abstraction génère un résultat plus compact, donc un très petit nombre de phrases; celles qui sont assez redondantes dans le texte. Notre objectif était alors d'intégrer les deux méthodes, et donc de résumer le texte une première fois par une méthode d'extraction, puis de résumer le résultat une seconde fois avec une méthode d'abstraction. Nous pourrions de cette façon, nous concentrer sur certaines phrases du texte uniquement, en filtrant une première fois les phrases les moins importantes, et ainsi, obtenir de meilleurs résultats par la méthode d'abstraction.

Nous allons tout d'abord, présenter certains exemples de résumés par les deux méthodes indépendamment, puis par la suite, nous allons expliquer les étapes de mise en commun des deux algorithmes, et des problèmes rencontrés dans la mise en place de notre approche.

L'extraction fonctionne très bien sur des textes et des articles, le document 1.a dans l'annexe est un extrait d'un article scientifique et est un exemple de texte à résumer par cet algorithme, et 1.b est le résultat obtenu par la méthode *LexRank*. On remarque que le résultat est satisfaisant; l'essentiel du texte est dévoilé dans le résumé, cependant, on remarque certaines erreurs minimes (des erreurs grammaticales, comme le début de phrase mis en valeur en rouge dans le document 1.b de l'annexe).

D'autre part, l'abstraction fonctionne uniquement sur des phrases très redondantes, le document 2.a dans l'annexe est un exemple de commentaires d'usager à résumer, et 2.b est le résultat obtenu par la méthode d'abstraction. On remarque pour cette méthode que c'est uniquement les phrases très redondantes qui sont sélectionnées.

Les deux algorithmes et leurs résultats étant présentés, on peut procéder à la mise en commun. À première vue, on pourrait penser que l'intégration des deux méthodes une à la suite de l'autre se ferait sans problème. Cependant, nous avons fait face à certains problèmes dans l'intégration. En effet, comme nous l'avons déjà dit précédemment, la méthode par extraction est utilisée pour génère des résume de texte ou article, alors que la méthode par abstraction est utilisée plutôt sur beaucoup de phrases très similaire et redondante. Les deux algorithmes opèrent alors sur des corpus très différents. La première tentative était de résumé des articles de journaux, le résultat de la méthode d'extraction était composé de plusieurs phrases de texte, qui était encore difficilement

traitable par la méthode par abstraction, puisqu'elles n'étaient pas assez semblables et redondantes. Nous avons alors pensé que ce serait mieux d'utiliser des commentaires d'usager plutôt qu'un texte, la méthode d'abstraction est prévue pour ce genre de texte, et la méthode *Lexrank* devrait identifier les phrases similaires sans problème. Cependant, les commentaires d'usager sont écrits avec un vocabulaire très diffèrent du vocabulaire utilisé dans les corpus d'entraînement que nous avons utilisé pour le calcul de *idf* (issue de documents *Wikipédia*). Ainsi, notre système n'était pas prévu pour traiter ce genre de texte. Avec cette approche, nous avons quand même réussi a générer des résultats acceptables, mais qui pourrait être nettement amélioré si le calcul des valeurs de *idf* était fait sur des corpus similaires aux textes traités (et donc que le corpus d'entraînement était similaire au corpus de tests, cependant, nous n'avons pas réussit à retrouvé un corpus d'entraînement de ce type, qui soit assez grand). Les résultats obtenus sont dévoilés dans les documents 3.a 3.b et 3.c de l'annexe. On remarque que le résumé obtenue par la méthode Lexrank relève des points important du corpus, alors que après avoir effectué la méthode d'abstraction sur le résultat, on obtient uniquement quelques phrases trés spécifique au texte.

Problèmes rencontrés

Lors de ce projet, nous avons rencontré plusieurs problèmes, certains mineurs, d'autres beaucoup plus gênants.

- Le premier est probablement le plus important : un mois après le début du projet, nous nous sommes aperçus que notre démarche avait un défaut majeur. En effet, nous comptions utiliser Opinosis pour résumer le résumé obtenu par LexRank. Cependant, LexRank ne crée aucune nouvelle phrase : il se contente d'extraire des phrases du texte. Ainsi, pour Opinosis, qui travaille sur la structure des phrases, tout ce qui peut s'obtenir à partir du résumé de LexRank peut

s’obtenir directement à partir du texte. Heureusement (si l’on peut dire), les deux algorithmes fonctionnent efficacement dans des situations très différentes : Opinosis pour résumer des opinions, et LexRank plutôt pour des articles. Ainsi, cumuler les deux permet d’avoir un algorithme fonctionnant pas trop mal dans les deux cas.

- Quelques soucis liés au fait qu’Opinosis n’était accessible qu’à distance, à partir de requêtes : impossible de, par exemple, mesurer la vitesse d’exécution. De plus, il arrive assez souvent (environ une fois sur deux) que la requête se solde par un timeout. Enfin, ce format impose une limite sur la taille des textes : impossible d’envoyer plus d’une centaine de lignes...
- Les deux algorithmes doivent travailler sur un ensemble de phrases plutôt que sur du texte brut. Nous avons donc dû implémenter un petit outil pour découper du texte en phrases, avec l’apparition du problème de : comment découper les phrases ? Dans un article typique, les phrases sont longues, avec beaucoup de virgules. Pour Opinosis, cela est très intéressant, mais malheureusement, LexRank choisit des phrases brutes : avoir de très longues phrases pose souci. Nous avons opté pour une solution intermédiaire, en ajoutant le point-virgule comme séparateur de phrases.
- Autre souci majeur : nous n’avons pas réussi à faire fonctionner ROUGE ! En effet, son utilisation demande l’installation de nombreuses librairies, et quelle que soit la machine sur laquelle nous avons testé, l’une d’elles pose problème. À nouveau, heureusement, si l’on peut dire, comme nos deux méthodes fonctionnent dans des cas différents, pas besoin de métrique très avancée pour déterminer si le résumé obtenu est bon ou pas : un simple coup d’oeil suffit.

Conclusion

Malheureusement, comme on vient de le voir, notre projet a été la source de nombreux problèmes de fond comme de forme. L'absence de métrique valable a probablement été le plus gros souci, et nous n'avons donc aucun résultat probant à montrer, autre que "ce résumé a l'air bien".

Nous avons cependant pu constater par nous mêmes les différences concrètes entre les deux méthodes : LexRank est beaucoup plus stable et fiable, car il y a dans tous les cas des résultats produits, bien que le résumé soit parfois peu naturel. Opinosis, par contre, a beaucoup plus de mal à produire des résultats : même dans un cas optimal (différents avis d'utilisateurs, par exemple), le résumé obtenu est souvent décevant, bien que l'on sente que l'idée derrière pourrait fournir d'excellents résultats avec un peu d'optimisation.

Si ce projet était à refaire, nous choisirions probablement une autre méthode par abstraction, éventuellement plus compliquée, où le code est disponible (ou au moins un exécutable, pour pouvoir mesurer les performances). Nous nous assurerions également de faire fonctionner notre métrique à l'avance pour avoir le temps, le cas échéant, de trouver une solution de secours.

Références:

Erkan, G. E., Radev, D. R. (2004). *LexRank: Graph-based Lexical Centrality as Salience in Text Summarization.* Tiré de http://www.jair.org/media/1523/live-1523-2354-jair.pdf
Opinosis : abstractive summarization. Tiré de http://dl.acm.org/citation.cfm?id=1873820

Annexe

Exemples partiels

In recent years, natural language processing (NLP) has moved to a very firm mathematical foundation. Many problems in NLP, e.g., parsing (Collins, 1997), word sense disambiguation (Yarowsky, 1995), and automatic paraphrasing (Barzilay & Lee, 2003) have benefited significantly by the introduction of robust statistical techniques. Recently, robust graphbased methods for NLP have also been gaining a lot of interest, e.g., in word clustering (Brew & im Walde, 2002) and prepositional phrase attachment (Toutanova, Manning, & Ng, 2004). In this paper, we will take graph-based methods in NLP one step further. We will discuss how random walks on sentence-based graphs can help in text summarization. We will also briefly discuss how similar techniques can be applied to other NLP tasks such as named entity classification, prepositional phrase attachment, and text classification.

Text summarization is the process of automatically creating a compressed version of a given text that provides useful information for the user. The information content of a summary depends on user's needs. Topic-oriented summaries focus on a user's topic of interest, and extract the information in the text that is related to the specified topic. On the other hand, generic summaries try to cover as much of the information content as possible, preserving the general topical organization of the original text. In this paper, we focus on multi-document extractive generic text summarization, where the goal is to produce a summary of multiple documents about the same, but unspecified topic. Extractive summarization produces summaries by choosing a subset of the sentences in the original document(s). This contrasts with abstractive summarization, where the information in the text is rephrased. Although summaries produced by humans are typically not extractive, most of the summarization research today is on extractive summarization. Purely extractive summaries often give better results compared to automatic abstractive summaries. This is due to the fact that the problems in abstractive summarization, such as semantic representation, inference and natural language

generation, are relatively harder compared to a data-driven approach such as sentence extraction. In fact, truly abstractive summarization has not reached to a mature stage today. Existing abstractive summarizers often depend on an extractive preprocessing component. The output of the extractor is cut and pasted, or compressed to produce the abstract of the text (Witbrock & Mittal, 1999; Jing, 2002; Knight & Marcu, 2000). SUMMONS (Radev & McKeown, 1998) is an example of a multi-document summarizer which extracts and combines information from multiple sources and passes this information to a language generation component to produce the final summary.

Document 1.a : Extrait d'un article scientifique à résumer par la méthode *Lexrank*

We will discuss how random walks on sentence-based graphs can help in text summarization.

Text summarization is the process of automatically creating a compressed version of a given text that provides useful information for the user.

On the other hand, generic summaries try to cover as much of the information content as possible, preserving the general topical organization of the original text.

In this paper, we focus on multi-document extractive generic text summarization, where the goal is to produce a summary of multiple documents about the same, but unspecified topic.

This contrasts with abstractive summarization, where the information in the text is rephrased.

Document 1.b : Résumé de 1.a par la méthode *Lexrank*

the bathroom was clean and the bed was comfy. the bathroom was clean and the bed was comfy. the bed was comfy and bathroom was clean. the bathroom was dirty. the bathroom was dirty. the bathroom was dirty. the bathroom was dirty. the bathroom was too dirty.

Document 2.a : Commentaire d'usager à résumer par la méthode d'abstraction (exemple fourni dans l'article)

the bathroom was dirty

the bathroom was clean

the bed was comfy

Document 2.b : Résumé de 2.a par la méthode d'abstraction

Exemple complet

the software is great .

transfering is easy , the software makes everythig pretty easy .

software is easy to use (although redhat software is better , but costs money).

so far the software for the pc works easily .

all in all this a great player that blows the i-pod away , with easy to use and understand features , software that is simple to understand and use , and a great sound (which is all that matters in a mp3 player) .

the software is very easy to use , and within 30 minutes of opening my box , i was out and about listening to any one of my 80 cds .

the software was great , as long as you have an adminstrative user account (windows xp) .

both are very easy to learn & use , and intuitive enough for even a novice .

i found using the supplied mediasource software very easy .

i had no problems setting up the software and getting my favorite cd 's transferred .

the player 's software is very easy to use and very good .

compared to musicmatch , the software has a better filing system and easier to use .

i 've had no problems with the software .

i found it intuitive to use (i did n't read any documentation for it and was using it successfully within a few minutes) .

no problem .

the software comes with plenty of defaults genres .
although the supplied software can be annoying at times , on whole it is excellent .
the software has n't been a hassle for me at all .
no crashes or problems .
the software installed flawlessly and without any problems on my windows 2k machine .
many people have written about the software being its one downfall , personally i have n't had a single problem with the software , i 'm running windows me .
the software runs smooth , it 's nice to look at , it 's very organized , easy to follow and makes things very simple .
the software could n't be better and the mp3 player works like a dream , i could n't give this anything but a 5 , i wish there was a 6 .
a lot of people complain about the creative software , but i actually find creative mediasource organizer to have some strengths over itunes .
third , creative software allows you to keep two windows open to look at both the content on the player and the computer .
the software is quick & easy to use & i found the entire process very easily mastered .
extremely convenient .
it was n't complicated to learn at all .
the new organizer software that ships with the nomad zen xtra could n't be easier to use .
given that i already have the zen the only reason i am happy with it is because of the notmad software .
many have complained about the software included but in my opinion it 's easy to use and effective .
great sound ; good interface ; replaceable , powerful battery ; good software ; wake-up , sleep timers ; multiple play modes
fact is , it could n't be easier to use .
if you use software correctly , put in all the id3 tags , you will get an awesome database collection , from which you can easily access your songs .

i dont know what other people are saying , but the software is awesome .
the software interface supplied was very easy to use but i went ahead and got red chair 's " notmad " software because of some advanced features i wanted .
it was no problem for me to use their software .
the main problem with the nomad jukebox zen xtra 30 gb is the software .
the interface software itself , which should be user-friendly , was anything but easy-to-use .
deficiencies with zennx are easily overcome with 3rd-party earphones ($ 20 +) and software ($ 25).
music match jukebox is n't the greatest , the search funtion is n't fast even when accessing it with the hotkey shortcut .
included creative software is pretty poor .
had there been strange noises , or obvious defects , i would have accepted a replacement , but this is clearly from a defective and inferior operating system and i for one do not intend on waiting for patches .
however , the creative software is not real intuitive .
the software is somewhat nice to look at , using it is not the easiest .
the software , which comes with player is not good at all .
so yes , as with all software bundled with creative products , it was useless and ugly .
d) software is another great misfortune -- hard to operate , crashes frequently , screwed my music library up , doesnt work with mp4s , the list goes on and on
the software failed repeatedly .
software downloads on the site crashed my pc .
the nomad jukebox zen xtra is a very good mp3 player but the software is it gets hurt .
on my main computer the software did not work right .
the software is crap .
otherwise , it is difficult to figure out and awkward .
but the major problem i had was with the software .
what really made this a mediocre mp3 player for me was the software .

my only reservations about this product concern the tagging process and the way it interacts with the software .
also , i can 't understand why the software does not ignore " the " when it lists the cds in alphabetical order .
there are a couple things i didnt like though but nothing serious : a little larger than other mp3s but still light , the software takes some time to get used to (maybe 10-15 mins) , and this thing would definitely be destroyed with one fall .
finally the software used to put all the songs into it is vastly inferior to itunes on two levels , first it is poorly integrated into the overall player and clunky to use especially with tagging and second the wma audio format sucks , flat out sucks in terms of sound quality compared with the vastly superior aac format the apple ipod uses and it sticks you with one of the sub par paid music services such as napster instead of allowing to use the itunes store , which , if you ever plan to pay for your music , is by far the best .
* software is absolutely terrible :
-- > you 'll be changing track information a lot because the cd database information utilized by the software is mediocre at best .
a good player at a great price with terrible software that makes ripping and transferring way more difficult than it should be .
my one issue with the mediasource software is that , despite pulling id3 tag data from the cdds service , the ripping process fails to populate the " year " field .
the only two things that stop me from giving it 5 stars are the ho-hum software that comes with the system , and the traveling case it comes with .
this player is an overall disappointment with a couple of big flaws that potential buyers should be wary of ; and all readers need to know about a third party software company that resolves virtually all of the creative mediasource file transfer issues .
1. creative mediasource software is ok to rip , catalog & burn music if you are starting from scratch ; but i used musicmatch to rip my cds before i bought the zen (278 cds ; 3,400 tracks ; 12.5 gb of music all ripped at 128bps) and guess what ?

the pc-side software can be goofy and takes a little getting used to but it 's not as bad as some reviewers have indicated .
the software that came with it was tough to load .
about 10 % of the time it can 't find the device when i attach it to my pc .
the software sucks .
software definitely sucks , hangs up my notebook about half the time , but i do n't mind buying notmad .
the creative software is awkward / difficult to use and just plain does n't load on some computers .
in fact , each of my creative mp3 players had software problem in the first place .
in short , this player is good at size , value , hardware design but really bad at its software
creative software stinks .
the software is not good either and the driver has trouble working with other sound blaster products , which is very dumb .
the controls are somewhat harder to use , and there are some oddities about the software tagging system , but most people can live with this .

Document 3.a : Commentaire d'usager à résumer par la méthode d'extraction suivi par la méthode d'abstraction

all in all this a great player that blows the i-pod away , with easy to use and understand features , software that is simple to understand and use , and a great sound (which is all that matters in a mp3 player) .
the software is very easy to use , and within 30 minutes of opening my box , i was out and about listening to any one of my 80 cds .
i had no problems setting up the software and getting my favorite cd 's transferred .
i 've had no problems with the software .
the software hasn't been a hassle for me at all .

many people have written about the software being its one downfall , personally i haven't had a single problem with the software , i 'm running windows me .
the software couldn't be better and the mp3 player works like a dream , i couldn't give this anything but a 5 , i wish there was a 6 .
the software is quick & easy to use & i found the entire process very easily mastered .
the new organizer software that ships with the nomad zen xtra couldn't be easier to use .
many have complained about the software included but in my opinion it 's easy to use and effective .
the software interface supplied was very easy to use but i went ahead and got red chair 's " notmad " software because of some advanced features i wanted .
d) software is another great misfortune -- hard to operate , crashes frequently , screwed my music library up , doesnt work with mp4s , the list goes on and onthe software failed repeatedly .
also , i can 't understand why the software does not ignore " the " when it lists the cds in alphabetical order .
finally the software used to put all the songs into it is vastly inferior to itunes on two levels , first it is poorly integrated into the overall player and clunky to use especially with tagging and second the wma audio format sucks , flat out sucks in terms of sound quality compared with the vastly superior aac format the apple ipod uses and it sticks you with one of the sub par paid music services such as napster instead of allowing to use the itunes store , which , if you ever plan to pay for your music , is by far the best .
software definitely sucks , hangs up my notebook about half the time , but i don't mind buying notmad .
the creative software is awkward / difficult to use and just plain doesn't load on some computers .

Document 3.b : Résumé de 3.a par la méthode *Lexrank*

the software is easy

software is great

easy to use

player is good

n't be easier to use

nice to look at

Document 3.c : Résumé de 3.b par la méthode *d'abstraction*