Universiteit Gent

Faculteit Ingenieurswetenschappen en Architectuur

Verbeterde aanbevelingssystemen op basis van content recognition in tekst

door

Bjorn VANDENBUSSCHE

Promotor: Prof. L. MARTENS

Scriptiebegeleider: Dr. ir. T. DE PESSEMIER

Scriptie ingediend tot het behalen van de academische graad van
Master of Science in de industriële wetenschappen: informatica

Academiejaar 2015–2016

Voorwoord

Graag had ik enkele mensen bedankt voor hun hulp en medewerking aan deze masterproef. In de eerste plaats gaat mijn dank uit naar mijn promotor, Luc Martens en naar mijn begeleider, Toon De Pessemier om mijn masterproef in goede banen te leiden.

Deze masterproef werd uitgevoerd bij de vakgroep informatietechnologie (Intec) van de Universiteit Gent. Ik wil hen graag bedanken voor de kansen die ze mij boden met deze masterproef.

Bjorn Vandenbussche, juni 2016

Toelating tot bruikleen

Bjorn Vandenbussche, juni 2016

Verbeterde aanbevelingssystemen op basis van content recognition in tekst

door

Bjorn VANDENBUSSCHE

Scriptie ingediend tot het behalen van de academische graad van
Master of Science in de industriële wetenschappen: informatica

Academiejaar 2015–2016

Promotor: Prof. L. MARTENS
Scriptiebegeleider: Dr. ir. T. DE PESSEMIER
Faculteit Ingenieurswetenschappen en Architectuur
Universiteit Gent

Vakgroep Informatietechnologie

Samenvatting

Het doel van dit onderzoek is om te achterhalen welke vormen van content recognition het aanbevelingssysteem kunnen helpen om nuttige aanbevelingen te leveren. Verschillende informatie-extractietechnieken kunnen ervoor zorgen dat het aanbevelingssysteem een beter beeld krijgt van de voorkeur van de gebruiker door werkelijk rekening te houden met de inhoudelijke aspecten van de items die aanbevolen worden.

Daartoe zijn verschillende vormen van content recognition uitgetest. Een eerst is classificatie. Door verschillende items binnen dezelfde categorie te classificeren kan de context al heel wat duidelijker afgeleid worden. Voor twee verschillende Nederlandse datasets werden met het WEKA framework de classifiers gezocht die de beste classificatie opleveren voor elke categorie.

Om vervolgens bepaalde *topics* binnen categorieën of samenhorende items over verschillende categorieën heen te vinden wordt clustering toegepast met het k-means en fuzzy k-means algoritme in Apache Mahout. Ter optimalisatie van deze clustering wordt gebruik gemaakt van named entity recognition om een beter beeld te krijgen van topics.

Door zowel de informatie van de classificatie als van de clustering te combineren krijgt het content-based aanbevelingssysteem een beter beeld van de inhoud van items en worden betere aanbevelingen gegenereerd.

Trefwoorden

text mining, big data, aanbevelingssystemen, machine learning, classificatie, clustering, named entity recognition

Better recommender systems based on content recognition in text

Bjorn Vandenbussche

Supervisor(s): Luc Martens

Abstract—**This article tries to extend the limits of content-based text recommender systems by exploiting content recognition in text. It proposes a new system that is able to classify and cluster text in order to get a better view on a users need.**

Keywords—**text mining, big data, recommender systems, machine learning, classification, clustering, named entity recognition**

I. Introduction

RECOMMENDER systems help individuals address the challenges of information overload on the Internet. Although a lot of content is available, it remains a though task for users to find the most relevant information. Classical keyword-based recommenders offer just a partial solution. Preferences of users, earlier experiences and already consumed content are not taken into account in typical search engines. That's why recommender systems can be a useful tool in order to select and consume content.

Classical recommender systems work based on consumption behavior (ratings, purchases, clicks), sometimes supplemented by information about content (meta data) like keywords, author, category, resume, etc. The problem with these systems is that the description isn't always perfect. Recommender systems won't work well in those cases, as they expect an exact match of meta data or at least an overlap in used words.

In this abstract we will first discuss what kinds of content recognition can help the recommender system to deliver useful recommendations. Different techniques for information extraction can make sure the recommender gets a better view on the preferences of a user, by really taking the content of items being recommended into account.

II. Concept

Text documents appear under all kinds of file formats (html, doc, pdf, etc). Each format has its own markup and somehow tries to upgrade regular text to something more useful in their own environment. Unfortunately this doesn't work well with content recognition systems, as they require plain text to work with. To address this problem all documents that are processed in this system first pass through Apache Tika [1]. This toolkit allows the extraction of both text and meta data from all kinds of text documents, making it possible for content recognition systems to consume the plain text of different sources. Using this extraction makes it possible to use this system for all kinds of text recommendation problems. The plain text produced by Tika will then be consumed by two systems of content recognition: classification and clustering.

By automatically classifying a text into a predefined structure of categories, we try to deduct context of that item. This classification allows us to gather similar documents and so recommend texts to users that they might like, considering they enjoyed some other content in the same category.

Clustering on the other hand allows to find clusters of similar texts (also referred to as topics). These clusters may be situated inside of an existing category (making classification more precise), but may as well group articles from way different categories since they have no knowledge of classification at all. They don't require any predefinition of structures, but try to calculate document similarity in an n-dimensional space.

By combining the information from both classification and clustering the recommender systems gets a better view on the content of the items and is thereby capable of generating better recommendations.

III. Content recognion

A. Preprocessing

The structure of plain text isn't sufficient to be efficiently processed by a computer. In order to comprehend what a text is about, it has to be preprocessed to a feature vector that can be used as input for content recognition systems. The requirements for preprocessing are similar for both classification and clustering. Yet for classification one additional step can be added to lower the dimensionality of the features.

A.1 Tokenization

The first step in preprocessing a text document is tokenization. This process makes sure a document gets split into a stream of words by removing all punctuation marks and replacing all non-textual characters by spaces. The set of different words from all documents gets merged into the dictionary of the document collection. This dictionary then gets used for both classifying or clustering new documents.

A.2 Filtering and stemming algorithms

In order to remove common stop words that have little or no meaning filtering is applied. Examples of removed words include: articles, conjunctions, prepositions, etc. These words tend to add noise to the feature vector, which is something we like to avoid.

Stemming algorithms try to transform a word to their word stem or some kind of root form of a word. It is used to make related words map to the same stem. This can be achieved by converting plural back to singular form or by converting verbs back to their word stem. These words aren't always morphologically correct, but are used for This is achieved by applying Porters algorithm for stemming [2].

A.3 Features selection

Feature selection is a step used to preprocess text for classification systems. It makes sure that irrelevant attributes get removed from the dictionary. In that way a subset of the original dictionary is generated. This mostly reduces processing power and search space required for the vector model. Feature selection is mostly interesting in text mining because of the high dimensionality of the features. A lot of those features turn out to be irrelevant to the classification problem, or can even be seen as noise for the classifier.

B. Classification

For two Dutch datasets (*Het Laatste Nieuws*, 2008 and Wikipedia) multiple classifiers from the WEKA framework [3] were researched. Both datasets are structured as a two-leveled tree. In order to find the optimal classification for each node of the tree, tests were conducted for the top two categories of the dataset. We first try to classify the text in one of the top categories and then classify it into a subcategory. This research solely wants to test what classifiers work best for these datasets, and doesn't describe the algorithms themselves.

B.1 Individual classifiers

The first dataset tested is Het Laatste Nieuws. For the first level, top classifiers include DMNBtext (89.86%), NaiveBayesMultinomial (88.75%), SMO (86.98%), ComplementNaiveBayes (86.45%) and Logistic (82.83%). These ratings are the percentages of texts they can classify correctly being trained with one month of data (Januari 2008, 10121 articles), using 10-fold cross validation.

For the second dataset (Wikipedia) training data gets generated with 1000 random articles chosen for each subcategory. The classification problem for this dataset seems to be a lot harder as the class structure is not as strictly defined as with the first dataset. Wikipedia tends to use an acyclic graph representation of articles, since they want to link multiple topics over different categories to inform a user of their encyclopedia. This results in much poorer classification. Top classifiers include ComplementsNaiveBayes (51.44%), NaiveBayesMultinomial (51.33%) and DMNBtext (50.0%).

B.2 Combining classifiers

By combining good individual classifiers it is possible to achieve better classification. Combination algorithms used are voting, stacking and grading. Results for the first dataset show a 0.6-0.12% improvement to classification precision for the first dataset. The beste combination algorithm found was for this dataset was stacking the top 4 classifiers and using linear regression to combine them (89.98%).

The second dataset also benefits from using a combination of classifiers. Best results are achieved by using a voting scheme (52.04% correctly classified).

C. Clustering

Clustering is used to group documents with a similar content. Each cluster contains some documents. It is important that documents inside of a cluster show big similarity (intra-cluster similarity), but differ with documents from other clusters (intra-cluster similarity). This makes up for a good measure of validity of a clustering.

C.1 Named entity recognition

With text documents, clusters tend to be related to certain named entities (NEs). To achieve better clustering natural language processing (NLP) gets added to the preprocessing using the Stanford Named Entity recognizer [4] [5]. The idea that NLP can help finding relevant clusters gets fueled by *(a)* the fact that NEs remain the same over different documents (it is difficult to paraphrase a name), *(b)* the frequency of NEs in a text shows its importance to the events being described in the text, and *(c)* two different stories containing the same NE most probably belong to the same cluster.

Because not all texts contain NEs it is very difficult to conduct clustering solely based on them. Because of that fact, in this system, NEs get an artificial higher term frequency than other (ordinary) words. Using this system interesting clusters get build with a good mixture of NEs and important keywords. On top of that, clusters (and thus recommendations) get more comprehensive for users of the recommender system.

C.2 K-means algorithm

In this system, Apache Mahout is used to process documents to a clustering by applying one of the best-known algorithms: k-means clustering. It tries to partition a given dataset $X = x_1, ..., x_n, x_n \in R^d$ in k different subsets (clusters) $C_1, ..., C_k$. It expects a certain value for the number of clusters k that should be constructed. This requirement is a big drawback for this algorithm. If one chooses a k that is too big, clusters will be too small and won't bring any relevance to users. If on the other hand k gets chosen too small, clusters will tend to be bigger and their topic will be much more general than we want it to be.

In order to overcome the issue of finding the optimal k, canopy clustering is used. The main idea with canopy clustering is that the calculations that are necessary in order to calculate distances can diminish if the dataset first gets divided in overlapping subsets. Distances can then be calculated between each pair of items that belong to the same subset. The strength of this algorithm lies in the speed by which clusters are formed. This strength also its largest weakness, as clusters that get formed by canopy clustering aren't always accurate. Luckily the output of canopy clustering (and so the optimal k) can be used as input for the k-means algorithm. This algorithm is able to find good and relevant clusters.

IV. Recommender system

The recommender system used to process the input from both the classification and clustering is content based (built with Lenskit [6]). It uses the information as tags and tries to find relevant items based on those tags. This recommender was compared to a TFIDF content based recommender. This baseline uses the contents of an item to find similar documents.

V. Conclusion

The recommender evaluation shows a vast improvement over other recommendation systems. By finding relevant items really

based on their contents, a recommender can help users finding and consuming more relevant content.

REFERENCES

[1] The Apache Software Foundation, "Apache Tika," .

[2] M.F. Porter, "An algorithm for suffix stripping," 1980.

[3] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten, "The WEKA data mining software," *ACM SIGKDD Explorations*, vol. 11, no. 1, pp. 10–18, 2009.

[4] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky, "The Stanford CoreNLP Natural Language Processing Toolkit," *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, vol. 53, pp. 55–60, 2014.

[5] Soto Montalvo, Raquel Martínez, Víctor Fresno, and Agustín Delgado, "Exploiting named entities for bilingual news clustering," *Journal of the Association for Information Science and Technology*, vol. 66, no. 2, pp. 363–376, feb 2015.

[6] Michael D. Ekstrand, Michael Ludwig, Jack Kolb, and John T. Riedl, "LensKit," *Proceedings of the fifth ACM conference on Recommender systems - RecSys '11*, p. 349, 2011.

Inhoudsopgave

Hoofdstuk 1

Inleiding

1.1 Bestaande situatie

De enorme hoeveelheid informatie op het internet confronteert de eindgebruiker met het probleem van overaanbod. Hoewel alle content beschikbaar is, is het voor de gebruiker moeilijk om de meest geschikte, de meest interessante content terug te vinden.

De klassieke sleutelwoord gebaseerde zoeksystemen bieden hierbij slechts een gedeeltelijke oplossing. Voorkeuren van de gebruiker, eerdere ervaringen, en reeds geconsumeerde content worden niet in rekening gebracht bij de klassieke zoeksystemen. Daarom zijn aanbevelingssystemen een nuttig hulpmiddel voor het ontdekken, selecteren en consumeren van content. Vele online diensten zoals Amazon [28] en Netflix [4] hebben reeds een aanbevelingssysteem dat zijn nut al meermaals bewezen heeft.

De klassieke aanbevelingssystemen werken op basis van consumptiegedrag (beoordelingen, aankopen, klikgedrag), soms aangevuld met informatie over de content (metadata) zoals sleutelwoorden, auteur, categorieën, samenvatting,... Het probleem bij deze systemen is dat de beschrijving niet altijd perfect is. Zo kunnen er schrijffouten optreden, synoniemen of hyponiemen voorkomen, of een ander lexicon of thesaurus gebruikt zijn.

Aanbevelingssystemen zullen in dat geval niet goed werken, gezien deze een exacte overeenkomst van de metadatavelden verwachten of op zijn minst een overlap in gebruikte woorden. Zo zullen bij klassieke aanbevelingssystemen twee synoniemen ten onrechte als twee verschillende concepten beschouwd worden. Bepaalde verbanden tussen gerelateerde content items zullen dus nooit door een klassiek aanbevelingssysteem ontdekt worden. Op het web is er echter meer

informatie beschikbaar dan typisch gebruikt wordt in aanbevelingssystemen. Door het begrijpen van de inhoud, kunnen aanbevelingssystemen meer nauwkeurige suggesties doen.

1.2 Doelstelling

Het doel van deze masterproef is de ontwikkeling van een aanbevelingssysteem dat gebaseerd is op content recognition in tekst. Via content recognition technieken kan informatie uit teksten gehaald worden die als input voor een aanbevelingssysteem gebruikt kunnen worden. Informatie-extractie kan ervoor zorgen dat het aanbevelingssysteem een beter beeld krijgt van de voorkeuren van de gebruiker door werkelijk rekening te houden met de inhoudelijke aspecten van de items die aanbevolen worden. Tevens zal content recognition ervoor zorgen dat de afhankelijkheid van metadata in aanbevelingssystemen minder sterk is, of zelfs verdwijnt.

1.3 Overzicht

In dit hoofdstuk werd reeds geschetst wat de doelstelling van deze masterproef is. In hoofdstuk 2 wordt verder ingegaan op zowel data mining, text mining als aanbevelingssystemen. Hoofdstuk 3 bevat een overzicht van de gebruikte datasets. In hoofdstuk 4 wordt de algemene architectuur geschetst van het ontwikkelde systeem, zonder daarbij de implementatiedetails te vermelden. Deze details komen verder aan bod in hoofdstuk 5. In hoofdstuk 6 wordt nog kort ingegaan op hoe het ontwikkelde aanbevelingssysteem omgaat met data. Ten slotte bevat hoofdstuk 7 een overzicht van de belangrijkste kenmerken van het systeem.

Hoofdstuk 2

Achtergrond

De opgeslagen informatie neemt elke dag enorm toe. Ontdekken van patronen en trends uit deze steeds groeiende data is een taak die steeds moeilijker wordt. Bepaalde technieken kunnen gebruikt worden om het probleem op te lossen. De basistechniek daarbij is *data mining*, waarvan toepassingen o.a. *text mining* en *web mining* zijn [34].

Dit hoofdstuk voorziet de nodige achtergrondinformatie om de werking van dergelijke text mining systemen te begrijpen. Eerst wordt data mining als algemeen, abstract begrip omschreven. Uiteraard bestaan er genoeg goede publicaties (o.a. [18]) die verder op dit onderwerp ingaan. Vervolgens komt zijn afgeleide toepassing, text mining, aan bod. De belangrijkste topics binnen dit domein die verband houden met deze masterproef worden ten slotte nader uitgelicht.

Om te begrijpen hoe een aanbevelingssysteem vervolgens kan omgaan met deze informatie wordt in dit hoofdstuk ook kort ingegaan op dat onderwerp. De verschillende technieken worden high-level beschreven en er wordt geduid welke techniek het meest geschikt is voor het aanbevelen van teksten aan gebruikers.

2.1 Data mining

Data mining werd in 1980 voor het eerst gebruikt om data om te zetten naar kennis. Het doel van data mining is om impliciete, vooraf onbekende trends en patronen uit databases te halen. Daarbij wordt gebruik gemaakt van verschillende technieken: classificatie, clustering, neurale netwerken, beslissingsbomen, etc.

Data mining is deel van het *knowledge discovering process* of *knowledge discovery in databases* (KDD). Dit proces bestaat uit verschillende stappen [11]:

1. Begrijpen van business: de objectieven en verwachtingen worden gedefinieerd.
2. Begrijpen van data: data wordt uit het data warehouse geselecteerd op basis van de gedefinieerde objectieven en verwachtingen.
3. Voorbereiden van data: de kwaliteit van de data wordt verbeterd.
4. Modelleren van data: een data mining algoritme wordt geselecteerd en toegepast op de data die voorbereid werd in de vorige stap.
5. Evaluatie: de relaties en patronen worden geanalyseerd en geldige patronen volgens de vooraf opgestelde doelen worden geselecteerd.
6. Visuele representatie: de kennis die ontdekt is wordt visueel voorgesteld. Deze resultaten kunnen opgeslagen en samengevoegd worden om de business vooruit te helpen.

2.2 Text mining

Content recognition of *text mining* verwijst naar de extractie van interessante informatie en kennis uit ongestructureerde tekst. Het probeert om de verborgen informatie te onthullen door middel van methodes die enerzijds met een groot aantal woorden en structuren in natuurlijke taal kunnen omgaan en anderzijds een zekere vaagheid en onzekerheid kunnen verwerken. Text mining kan naast het werken met gestructureerde data (zoals de data die in data mining verwerkt wordt) ook werken met ongestructureerde of semi-gestructureerde data zoals e-mails, volledige tekstdocumenten, HTML bestanden, etc. Text mining wordt daarom beschreven als een interdisciplinaire methode op basis van *information retrieval*[1], *machine learning*, statistiek, computationele taalkunde en vooral data mining [20].

Om grote collecties van documenten te verwerken, moeten tekstdocumenten vooraf verwerkt worden om de informatie op te slaan in een datastructuur die beter geschikt is dan een tekstbestand. De meeste text mining-methodes gaan ervan uit dat een tekstdocument voorgesteld

[1]Het vinden van documenten die antwoorden bevatten op vragen, maar niet het vinden van de antwoorden zelf.

kan worden als een set van woorden (*bag-of-words* representatie, zie 5.1.2). Ondertussen bestaan echter methodes die proberen om de syntactische structuur of de semantiek in de tekst uit te buiten. Op die manier kan de belangrijkheid van een woord achterhaald worden. Hiervoor wordt vaak een vector representatie gebruikt, die voor elk woord een numeriek gewicht bijhoudt. Enkele belangrijke modellen die hiervoor gebruikt worden zijn het vector space model [41], het probabilistische model [40] en het logische model [39]. Het vector space model wordt verder besproken in 5.1.3.

2.3 Data mining methodes voor tekst

De hoofdreden om data mining methodes in te zetten voor documentencollecties is om ze te structureren. Een structuur kan het gemakkelijker maken voor de gebruiker om een documentencollectie te raadplegen. Bestaande methodes die toestaan om documentencollecties te structureren proberen om kernwoorden aan documenten te koppelen op basis van een gegeven set kernwoorden (via classificatie of categorisatie, zie 2.3.1) of proberen de documentencollectie automatisch te structureren in groepen van gelijkaardige documenten (*clustering*, zie 2.3.2). Om dit te bekomen doet men soms beroep op natural language processing en informatie-extractie (zie 2.3.3).

2.3.1 Classificatie

Classificatie of categorisatie heeft als doel om voorgedefinieerde klasses toe te kennen aan tekstdocumenten. Het is een *supervised* techniek die de *classifier* traint op basis van gekende voorbeelden en zijn opgesteld model vervolgens gebruikt om ongekende voorbeelden automatisch te categoriseren [34].

Om classificatie te evalueren wordt gebruik gemaakt van drie verschillende waarden [37]: *precision*, *recall* (vangst) en *F-score*. Om deze waarden in perspectief te plaatsen is het nodig om de *confusion matrix* te introduceren in 2.1.

Daarbij staan p en n respectievelijk voor positief en negatief en zijn p' en n' de voorspelde waarden voor p en n door een bepaalde classifier. Daarbij kan een geval True Positive (TP), False Positive (FP), False Negative (FN) of True Negative (TN) zijn. Met deze gegevens kan een zinvolle definitie gegeven worden aan precision als de proportie van voorspelde positieve gevallen die echt positief zijn.

Tabel 2.1: Voorbeeld van een confusion matrix

	p	n
p'	TP	FP
n'	FN	TN

$$Precision = \frac{TP}{TP + FP} \tag{2.1}$$

Recall wordt gedefinieerd als de proportie echte positieven die juist voorspeld worden.

$$Recall = \frac{TP}{TP + FN} \tag{2.2}$$

Ten slotte kan de F-score opgesteld worden als harmonisch gemiddelde van de twee:

$$F - score = 2 \times \frac{precision \times recall}{precision + recall} \tag{2.3}$$

2.3.2 Clustering

Clustering is een *unsupervised* techniek waar geen patronen voorgedefinieerd zijn. De methode is gebaseerd op een concept dat gelijkaardige documenten of teksten in dezelfde cluster groepeert. Elke cluster bevat dus een aantal documenten. De clustering wordt als beter beschouwd indien de inhoud van de documenten intra-cluster meer gelijkheid vertoont dan de inhoud van de documenten inter-cluster.

Clustering wordt gebruikt om gelijkaardige documenten te groeperen, maar verschilt van classificatie omdat documenten bij clustering on-the-fly ingedeeld worden in clusters, in plaats van in vooraf gedefinieerde klasses of topics.

Types van clustering

Een volledige collectie clusters wordt vaak gerefereerd als een *clustering* [46]. Er bestaan verschillende types van clustering: hiërarchisch versus partionele clustering, exclusieve versus *overlapping* versus fuzzy clustering en complete versus partiële clustering.

- Hiërarchisch versus partioneel: een partionele clustering is een verdeling van de dataset in niet-overlappende subsets (clusters) zodat elk object exact in één cluster terecht komt. Als we toestaan dat clusters ook subclusters bevatten, dan krijgen we een hiërarchische clustering. De clusters zijn hierbij georganiseerd als een boom. Elke knoop in de boom (behalve de bladeren) is een unie van al zijn kinderen. De root node bevat dus alle objecten.

- Exclusief versus overlapping versus fuzzy: een clustering die alle objecten aan één cluster assigneert heet een exclusieve clustering. Er zijn echter veel situaties waar items in meerdere clusters kunnen geplaatst worden, waardoor ze beter gebaat zijn bij een niet-exclusieve clustering (overlapping clustering). Bij een fuzzy clustering behoort elk object tot een cluster met een gewicht tussen 0 (behoort absoluut niet tot) en 1 (behoort er absoluut wel tot). Meestal wordt als restrictie gebruikt dat de som van alle objecten gelijk moet zijn aan 1. In praktijk wordt de fuzzy clustering vaak geconverteerd naar exclusieve clustering door elk object aan de cluster toe te wijzen waar zijn gewicht het grootst is.

- Compleet versus partieel: een complete clustering voegt elk object toe aan een cluster, terwijl partieel clusteren dit niet doet. De motivatie voor een partiële clustering is dat objecten in een dataset soms niet tot een gedefinieerde groep behoren. Denk hierbij aan uitschieters of ruis.

2.3.3 Informatie-extractie

Natuurlijke taal bevat meer informatie die niet direct geschikt is voor analyse door een computer. Toch kunnen computers door grote hoeveelheden tekst gaan en mogelijk interessante informatie blootleggen. Dit kan gaan van woorden tot zinnen of zelfs tot volledige passages tekst. Informatie-extractie [50] wordt daarom gezien als een begrensde vorm van volledig taalbegrip, omdat we vooraf weten naar welke semantische informatie we zoeken. De hoofdtaak is om stukken tekst te extraheren en daar specifieke attributen aan toe te kennen.

Een voorbeeld van informatie-extractie is *named entity extractie* [8]. Dit kan bekeken worden als een tagging-probleem gebaseerd op woorden. Het woord dat de entiteit start krijgt tag “B” (Begin), het volgende woord tag “I” (Inside) en de woorden buiten de entiteit “O” (Outside). Op die manier wordt een sequentieel classificatieprobleem opgesteld voor de labels van elk woord, met de omliggende woorden als feature vector.

2.4 Aanbevelingssytemen

Er bestaan verschillende strategieën om aanbevelingen te genereren. Deze strategieën worden onderverdeeld in volgende categorieën [2]:

- Content-based aanbevelingen: de gebruiker krijgt aanbevelingen gelijkaardig aan de items die hij vroeger prefereerde.
- Collaborative filtering: de gebruiker krijgt aanbevelingen die andere gebruikers met gelijkaardige smaak en voorkeur goed vonden.
- Hybride systemen: deze methodes combineren content-based aanbevelingen en collaborative filtering technieken.

Voor teksten wordt klassiek een content-based aanbevelingssysteem gebruikt. Verschillende soorten van metadata (sleutelwoorden, auteur, samenvatting, ...) worden dan gebruikt om aanbevelingen te genereren. Dit soort van systemen valt of staat dan ook met een correcte labeling van items. Enkel op die manier is het systeem in staat om items op een juiste manier te vergelijken.

Deze content-based systemen hebben verschillende voor- en nadelen [5].
Voordelen zijn:

- Onafhankelijk van de andere gebruikers: een content-based recommender maakt enkel gebruik van ratings om zijn eigen profiel op te bouwen. Bij collaborative filtering worden ook ratings van andere personen in rekening gebracht die gelijkaardige interesses hebben bij het aanbevelen van items.
- Transparant: het is eenvoudig uit te leggen hoe deze aanbevelingen werken. Door de eigenschappen van een item op te lijsten die ervoor gezorgd hebben dat de recommender dit item aan jou aanbeveelt kun je eenvoudig begrijpen waarom dit voor jou relevant is. Die lijst kan ook gebruikt worden om te beslissen of een item al dan niet betrouwbaar is.
- Nieuwe items: items die nog door geen enkele gebruiker een rating kregen, kunnen toch door content-based recommenders worden aanbevolen.

Aan content-based recommender hangen ook enkele nadelen:

- Gelimiteerde analyse van de inhoud: content-based technieken hebben een limiet op het aantal en het type van features die gebruikt worden. Soms is kennis van het domein nodig om te begrijpen waar een item precies over gaat.
- Overspecialisatie: er bestaat geen methode om iets "onverwachts" te vinden. Het systeem suggereert enkel items waar de score gelijkaardig is aan jouw gebruikersprofiel. Dit kan ervoor zorgen dat de recommender enkel nog items kan aanbevelen die reeds gezien zijn door de gebruiker.
- Nieuwe gebruikers: bij een nieuwe gebruiker moeten eerst ratings verzameld worden zodat een content-based recommender kan begrijpen wat de gebruiker goed vindt. Voor een nieuwe gebruiker kunnen bijgevolg geen betrouwbare aanbevelingen gedaan worden. Dit fenomeen wordt soms beschreven als het cold-start probleem.

Hoofdstuk 3

Datasets

Om het systeem te testen wordt gebruik gemaakt van twee datasets: Het Laatste Nieuws en Wikipedia. Beide documentencollecties zijn opgesteld in het Nederlands. Voor elke dataset wordt telkens de verdeling en het totale aantal artikels vermeld.

3.1 Het Laatste Nieuws

De datasets die in eerste instantie gebruikt zullen worden zijn afkomstig van Het Laatste Nieuws[1], een populaire online en offline krant in België. De artikels die gebruikt worden komen van de online krant en werden gedownload via het archief. Om rekenkracht te besparen tijdens het onderzoek worden twee datasets gebruikt; een dataset met alle artikels van januari 2008 (10126 artikels) en een dataset met alle artikels van het jaar 2008 (82753 artikels). De dataset van januari 2008 doet tevens dienst als trainsset voor de verschillende classifiers in sectie 2.3.1.

Elk artikel behoort toe aan een boomstructuur van categorieën met twee niveaus. Het aantal hoofdcategorieën varieert tussen elf en twaalf (tijdens 2008 werd 1 hoofdcategorie toegevoegd in vergelijking met januari 2008). Voor de dataset van januari 2008 kunnen de hoofdcategorieën met hun verdeling teruggevonden worden in tabel 3.1. De dataset voor het volledige jaar 2008 kan teruggevonden worden in tabel 3.2. Merk op dat de categorie "Planet"werd geïntroduceerd (aandeel 1.36%).

De data werd opgehaald met een zelfgeschreven Perl[2] scraper gekoppeld aan de RSS-feed van

[1] http://hln.be

[2] https://www.perl.org/

Tabel 3.1: Verdeling van artikels over de verschillende hoofdcategorieën voor dataset Het Laatste Nieuws, januari 2008

Categorie	Aantal	Gewicht
Nieuws	3209	31.69%
You	2952	29.15%
Sport	1381	13.64%
Geld	918	9.07%
Showbizz	575	5.68%
Reizen	369	3.64%
Bizar	353	3.49%
iHLN	128	1.26%
Wetenschap	114	1.13%
Auto	99	0.98%
Muziek	28	0.28%
Totaal	**10126**	**100.00%**

Het Laatste Nieuws. Deze selecteert de body van elk artikel m.b.v. een reguliere expressie. Deze documenten worden vervolgens opgeslagen onder hun respectievelijke categorie, zodat ze bruikbaar zijn als input voor content recognition systemen.

3.2 Wikipedia

Wikipedia[3] is een gemeenschapsproject met als doel in elke taal een complete encyclopedie op het internet te creëren. Om deze dataset op te stellen werden voor elke hoofdcategorie 10000 artikels gedownload, die evenredig verdeeld zijn over de onderliggende subcategorieën. Dit maakt dat er in totaal 90000 artikels zijn. Wikipedia heeft een veel grotere boomstructuur met meer subcategorieën dan Het Laatste Nieuws. Om de vergelijking met de andere dataset mogelijk te maken, werden de artikels gecategoriseerd onder de bovenste twee niveaus. Op die manier krijgt elk artikel op Wikipedia een indeling in een hoofd- en subcategorie.

Om bepaalde testen uit te voeren wordt nog een tweede subset (trainingsset) gemaakt van deze dataset, waarbij telkens 1000 artikels per hoofdcategorie opgenomen zijn in de dataset (en dus

[3] https://nl.wikipedia.org/

Tabel 3.2: Verdeling van artikels over de verschillende hoofdcategorieën voor dataset Het Laatste Nieuws, jaar 2008

Categorie	Aantal	Gewicht
Nieuws	35436	42.82%
Sport	16168	19.54%
Showbizz	6687	8.08%
Geld	6684	8.08%
You	5739	6.94%
Bizar	4616	5.58%
Reizen	2670	3.23%
iHLN	1141	1.38%
Planet	1128	1.36%
Auto	1010	1.22%
Wetenschap	1008	1.22%
Muziek	466	0.56%
Totaal	**82753**	**100.00%**

in totaal 9000 artikels).

Ook de Wikipedia dataset werd opgehaald met een zelfgeschreven Perl scraper. Deze haakt in op de API die wikipedia aanbiedt om de artikels relevant voor bepaalde categorieën op te halen.

Hoofdstuk 4

Architectuur van het ontworpen systeem

In dit hoofdstuk wordt de architectuur van het systeem beschreven zonder concreet in te gaan op de implementatie. Het doel is om de ontwerpkeuzes te verdedigen. Details in verband met de werking komen in hoofdstuk 5.

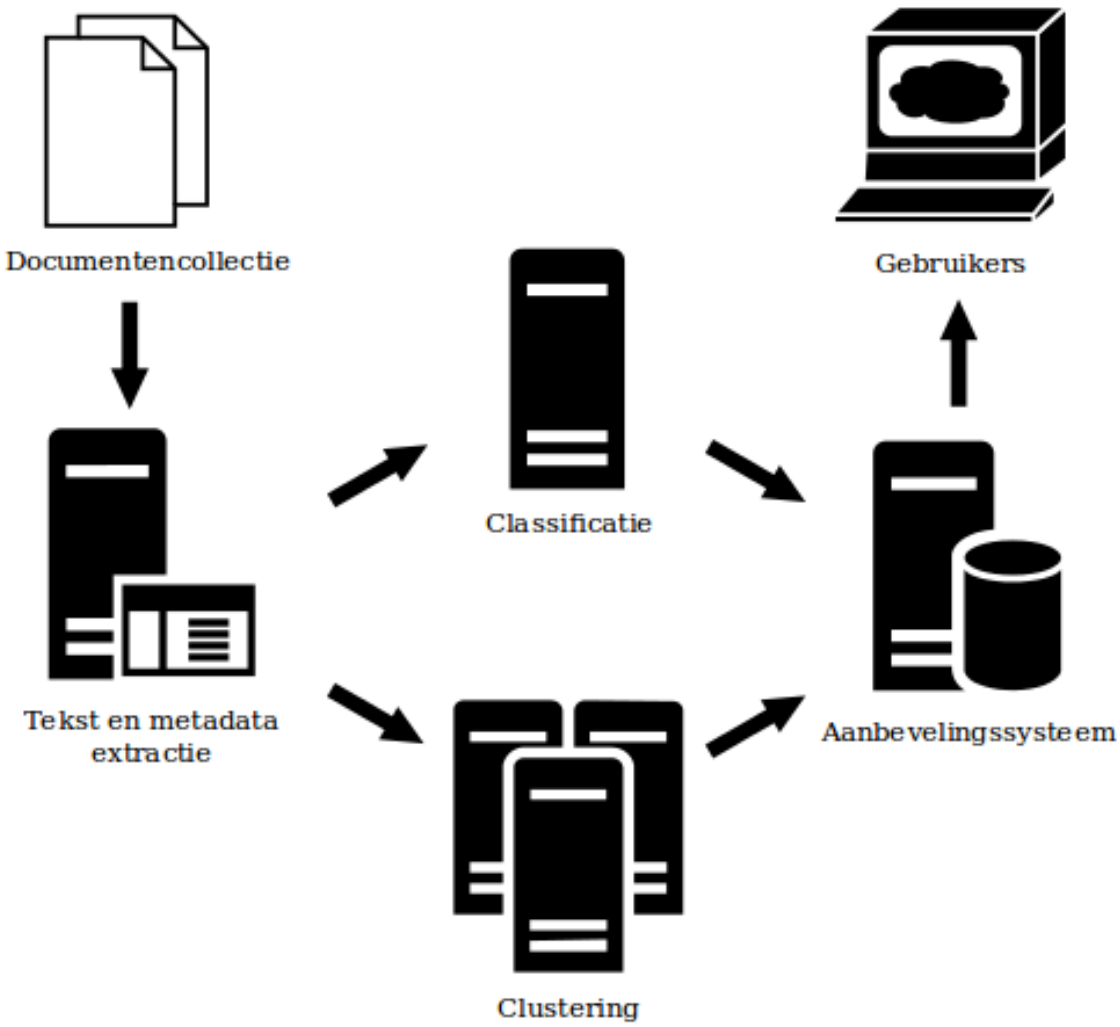
4.1 Algemeen overzicht

Figuur 4.1 geeft een high-level overzicht van de architectuur van het volledige systeem. Alle onderdelen van het ontworpen systeem zijn onderling onafhankelijk van elkaar. Dit maakt een robuuste, gedistribueerde eenheid waaraan relatief eenvoudig nieuwe componenten kunnen worden toegevoegd. Elk onderdeel wordt in de volgende paragrafen besproken.

4.1.1 Tekst en metadata extractie

Tekstuele data kan verschijnen onder verschillende vormen: HTML-pagina's, pdf-documenten, Word-documenten, etc. Elk van deze vormen heeft zijn eigen vorm en opmaak om de representatie als het ware up te graden van platte tekst naar iets wat bruikbaar is binnen hun eigen omgeving. Zo worden bij HTML verschillende tags toegevoegd om de opmaak te verfraaien of wordt er bij pdf-documenten een fixed-layout gemaakt om het document er onder alle omstandigheden hetzelfde te laten uitzien.

Figuur 4.1: High-level overzicht van de architectuur



Om met deze verschillende soorten van tekst te kunnen werken binnen een systeem van content recognition is het belangrijk om de platte tekst uit deze verschillende documentenformaten te extraheren. Op die manier wordt een gemeenschappelijke basis gecreëerd die als invoer zal dienen voor de verdere verwerking die dan niet meer afhankelijk is van het type van het document.

Naast het extraheren van tekst wordt in deze fase ook bepaalde metadata verzameld die voor alle documenten geldt. Zo kan de taal van een documentencollectie bepaald worden. Die taal kan dan gebruikt worden om specifieke verwerking voor die taal toe te passen.

4.1.2 Classificatie

Als we een bepaalde tekst automatisch in een structuur van categorieën kunnen situeren, krijgen we al heel veel informatie over mogelijke teksten die een gelijkaardige inhoud hebben. Teksten

die uit dezelfde categorie komen zullen de gebruiker sneller aanspreken en zijn dus mogelijk nuttig om aan te bevelen.

Voor classificatie is een vooraf bekende structuur nodig. Op basis van een leerverzameling wordt een *classifier* getraind om items onder bepaalde categorieën te classificeren. Het is dus de bedoeling om deze classifiers zo te kiezen dat voor elke categorie een maximaal aantal items juist geclassificeerd kan worden. Daartoe zal vooraf voor elke tak in de boomstructuur van een dataset een optimale classifier bepaald worden om zo tot een goed en betrouwbaar resultaat te komen.

4.1.3 Clustering

Omdat sommige categorieën groter zijn dan andere, bieden ze niet altijd evenveel informatie over de teksten die ze bevatten. Als we naar een dataset met nieuws kijken zijn artikels die gecategoriseerd worden onder "binnenlands nieuws" wel verwant, maar toch zitten daaronder heel veel topics die door het klassieke categoriseren niet kunnen gevonden worden. Denk maar aan politiek, onderwijs, religie, politie, etc.
Ook is het mogelijk dat sommige teksten uit een bepaalde categorie toch verwant zijn aan een tekst uit een totaal verschillende categorie. Zo kan Bart De Wever de ene dag in het nieuws komen als burgemeester van Antwerpen, maar de andere dag verschijnen in een quiz op tv. De artikels van de quiz zullen hoogstwaarschijnlijk niet onder nieuws gecategoriseerd worden. Maar toch zou het voor een gebruiker die een artikel over het burgemeesterschap van Bart De Wever leest interessant zijn om te zien dat hij ook in een bepaalde quiz meedeed.

Uit deze korte voorbeelden halen we twee punten die de meerwaarde van clustering tegenover classificatie blootleggen. Enerzijds worden bij classificatie de topics of onderwerpen binnen een klasse verwaarloosd en als één geheel gezien, anderzijds worden over de verschillende klasses heen geen gelijkaardige teksten gevonden. Door het toepassen van clustering op de documentencollectie worden op die manier topics gevonden die een nieuwe indicatie geven of een bepaalde tekst al dan niet moet worden aanbevolen aan een gebruiker.

4.1.4 Aanbevelen

De informatie die verzameld werd door de content recognition systemen wordt verzameld in een databank die door het aanbevelingssysteem kan geraadpleegd worden. Door de resultaten van

zowel de classificatie (situering van een item binnen de boomstructuur) als van clustering (het toebehoren aan een eventuele topic of onderwerp) te combineren kan de recommender op een eenvoudig manier gelijkaardige documenten zoeken. Deze recommender zal op basis van een content-based algoritme proberen een relevante aanbeveling te geven aan de gebruiker.

Hoofdstuk 5

Overzicht werking

In het vorige hoofdstuk werd de architectuur van het systeem geschetst. In dit hoofdstuk wordt verder ingegaan op de effectieve implementatie van het systeem en de eventueel andere overwogen opties. Voor alle back-end oplossingen werd gekozen voor de programmeertaal Java[1].

Alle testen werden uitgevoerd op een desktop computer met het Linux Mint besturingssysteem en met een Intel Core i5-2500k CPU (3,3Ghz), 8GB DDR3 RAM en 256GB HDD.

5.1 Voorbereiden van de dataset

5.1.1 Verwerking met Apache Tika

Om met verschillende documentencollecties te kunnen werken binnen het ontworpen systeem, wordt gebruik gemaakt van Apache Tika [47]. Tika laat toe om metadata van een tekst af te leiden. Die metadata kan o.a. de taal of het formaat van het document in kwestie zijn. Deze data kan belangrijk zijn bij de keuze van het stemmingsalgoritme dat gebruikt moet worden (dit is afhankelijk van de taal, zie 5.1.2) of laat toe om verschillende soorten documentformaten (pdf, doc, docx, csv, html, etc) om te vormen tot platte tekst zonder franjes. Deze tekst kan veel beter geïnterpreteerd worden door een computer.

Het parsen van een document bestaat uit twee delen. Het eerste deel is de taaldetectie, welke door de standaard *LanguageIdentifier* van het Tika framework wordt uitgevoerd. Het tweede deel is het effectief parsen van de tekst. Hiervoor wordt de *AutoDetectParser* van Tika gebruikt.

[1] http://www.java.com/

Deze kiest voor elk document de parser die voor dat type het beste resultaat zal opleveren. De resultaten van de parser komen in de vorm van JSON. Dit principe wordt in listing 5.1 geïllustreerd a.d.h.v. een item van Het Laatste Nieuws, waarbij de tekst werd doorgegeven als html-bestand. Zowel de taal als de tekst wordt perfect uit het html-bestand gehaald.

Listing 5.1: bjorn


```
{
"language":"nl",
"title":"5000 deelnemers opname clip klimaatsverandering\n",
"body":"Op het Klein Strand in Oostende zijn vanmiddag naar schatting 5.000 mensen samengekomen op
    een klimaathappening die onderdeel was van de internationale campagne 'The Big Ask'. De deelnemers
     figureerden in een filmpje van regisseur Nic Baltazar en Friends of the Earth vzw. Filmpje Youtube Op
    het Klein Strand vormden de deelnemers letters en slogans die vanuit de hoogte werden opgenomen en
    later zullen te zien zijn in een filmpje op Youtube. De actie 'Sos Klimaat : bouw een dijk tegen
    klimaatsverandering' kreeg de steun van talrijke BV's die als dj de massa animeerden. Onder meer
    Zohra, Flip Kowlier, Gabriel Rios en Adriaan Van den Hoof verleenden op het podium hun medewerking
    .Politici wakkerschudden Initiatiefnemer Nic Baltazar was in elk geval tevreden met de opkomst en de
    opnames en verwacht dat de clip op Youtube voor een half miljoen hits zal zorgen. Hij hoopt dat de
    actie en de clip politici zal wakker schudden omtrent de problematiek. (belga/ep)\n"
}
```


5.1.2 Preprocessing van tekstdocumenten

De platte tekst in de documentencollectie bevat op zich te weinig structuur om efficiënt verwerkt te worden door een computer. Om de data beter te kunnen begrijpen moeten we de dataset eerst omvormen tot een feature vector die vervolgens als input kan dienen voor content recognition systemen. De noden naar preprocessing zijn voor zowel classificatie als clustering zeer gelijklopend. Toch wordt voor classificatie een extra stap toegevoegd die probeert de dimensionaliteit van de features te verlagen (zie 5.1.4).

Tokenization

Om alle woorden te verkrijgen die gebruikt worden in een bepaalde tekst, wordt een *tokenization* proces toegepast. Dit proces zorgt ervoor dat een tekstdocument gesplitst wordt in een stroom van woorden door alle leestekens te verwijderen en alle tabs en niet-tekstuele karakters

door spaties te vervangen. De set van verschillende woorden uit alle tekstdocumenten wordt samengevoegd tot het woordenboek van de documentencollectie.

Het is echter ook mogelijk om een N-Gram tokenizer [31] te gebruiken. Deze geeft de mogelijkheid om eventueel twee of drie woorden samen te nemen en die als één woord in onze documentencollectie te zien. Dit kan voordeel geven bij o.a. personennamen, welke vaak bestaan uit twee woorden die voor een tekst belangrijk zouden kunnen zijn. Uiteraard wordt een eigennaam ook opgepikt indien alle woorden apart beschouwd worden.

Om de grootte van het woordenboek en dus de dimensionaliteit van de beschrijving van de documentencollectie te verkleinen, wordt de set van woorden verder gereduceerd door het toepassen van filters of *stemming*salgoritmes.

Filter- en stemmingsalgoritmes

Filters verwijderen woorden van het woordenboek en dus uit de documenten. De filtering die het meest wordt toegepast op tekstuele collecties is stopwoordfiltering. De stopwoorden die verwijderd worden zijn woorden die weinig of geen inhoud hebben. Voorbeelden zijn lidwoorden, verbindingswoorden, voorzetsels, etc.
Stemmingsalgoritmes proberen een woord om te vormen tot de standaardvorm van dat woord. Dit doen ze bijvoorbeeld door meervouden van zelfstandige naamwoorden naar het enkelvoud om te zetten of door werkwoorden naar hun stam te vereenvoudigen. Het stemmingalgoritme dat hier gebruikt wordt is Porters stemming algoritme voor de Nederlandse taal [26]. Het is een implementatie van Porters stemming algoritme [36], dat origineel enkel voor de Engelse taal werd ontworpen.

5.1.3 Vector space model

Ondanks de simpele datastructuur zorgt het vector space model ervoor dat grote collecties documenten efficiënt kunnen geanalyseerd worden. Het representeert documenten als vectors in een m-dimensionale ruimte. Elk document d uit de documentencollectie is beschreven als een numerieke *feature vector* $w(d) = (x(d, t_1), ..., x(d, t_m))$ waarbij $T = \{t_1, ..., t_m\}$ het woordenboek voorstelt. De hoofdtaak van de vector space representatie van documenten is het vinden van een geschikte encodering van de feature vector.

Elk element van de vector representeert meestal een woord (of groep van woorden) van de documentencollectie. De simpelste manier om een document te encoderen is om *binary term* vectoren te gebruiken. Als een woord voorkomt in het document wordt het corresponderende element op één gezet, komt het niet voor dan is het nul. De encodering wordt zo herleid tot een simpele Booleaanse vergelijking. Hierbij wordt de belangrijkheid van elk woord als gelijkwaardig beschouwd.

Om de performantie te verbeteren kunnen *term weighting schemes* worden gebruikt [42]. Het gewicht dat toegekend wordt aan een woord reflecteert de belangrijkheid of relevantie van dat woord in een specifiek document of collectie. Een woord met hoge frequentie in bepaalde documenten, maar dat weinig of niet voorkomt in de volledige documentencollectie, wordt een groot gewicht toebedeeld. Een gewicht $w(d,t)$ voor term t in document d wordt berekend als de term frequency $tf(d,t)$ vermenigvuldigd met de inverse document frequency $idf(t)$ - gedefinieerd als $idf(t) = \log \frac{N}{n_t}$). Dit beschrijft de specifiteit van een bepaalde term in een documentencollectie.

Naast term frequency en inverse document frequency wordt een normalisatie toegepast om ervoor te zorgen dat alle documenten dezelfde kans hebben om gevonden te worden, onafhankelijk van hun lengte. Deze techniek heeft zijn nut reeds bewezen in de praktijk .

$$w(d,t) = \frac{tf(dt) \log \frac{N}{N_t}}{\sqrt{\sum_{j=1}^{m} tf(d,t_j)^2 (log(\frac{N}{n_{t_j}}))^2}} \tag{5.1}$$

Hierbij stelt N de grootte van de documentencollectie D voor en is n_t het aantal documenten in D dat term t bevat.

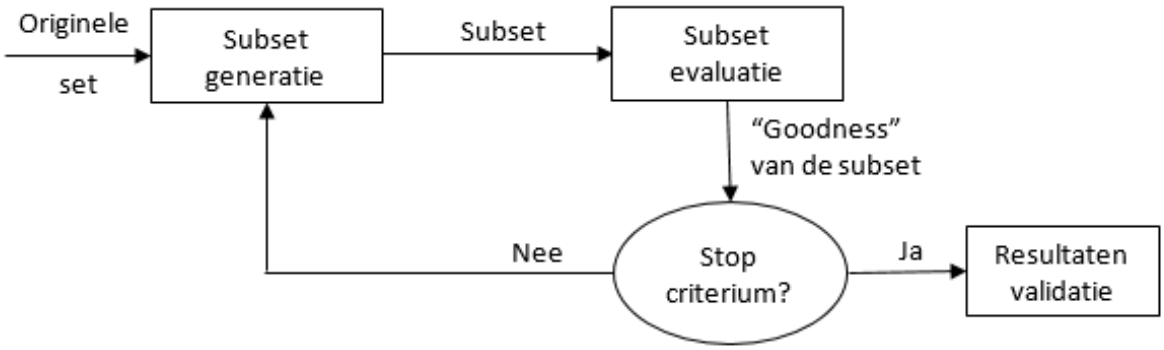
5.1.4 Feature selection

Feature selection of attribute selection is een proces dat enkel bij classificatie doorlopen wordt. Het zorgt ervoor dat attributen die niet relevant zijn verwijderd worden. Zo wordt een subset van de features gezocht die relevant zijn voor het doelconcept. Dit zorgt er o.a. voor dat de dataset kleiner wordt, waardoor minder rekenkracht en zoekruimte vereist is bij de effectieve verwerking van het vector model. Volgens [21] is het één van de meest belangrijke en meest gebruikte technieken voor data preprocessing bij data mining. Het reduceert het aantal features, het verwijdert irrelevante of redundante data en het zorgt er bijgevolg voor dat data mining algoritmes sneller werken. Het zorgt er o.m. ook voor dat de voorspelde accuraatheid en begrijpbaarheid van de resultaten verbeterd wordt. Feature selection is vooral interessant bij

text mining, omdat er een hoge dimensionaliteit is van de features en er veel irrelevante features voorkomen.

Als de dimensionaliteit van een domein bovendien vergroot wordt, verhoogt het aantal features N. Een optimale subset van features vinden is zeer moeilijk en veel problemen gerelateerd aan feature selection zijn NP-moeilijk. Volgens [21] bestaat een typisch feature selection proces uit vier stappen (zie figuur 5.1); subset generatie, subset evaluatie, stopcriterium en resultatenvalidatie.

Figuur 5.1: Vier stappen in feature selection



Subset generatie is een zoekprocedure die kandidaat feature subsets produceert voor evaluatie met een bepaalde zoekstrategie. Elke kandidaat subset wordt geëvalueerd en vergeleken met de vorige beste volgens een bepaald criterium. Als de nieuwe subset beter is, wordt de vorige beste subset vervangen. Dit proces wordt herhaald tot een bepaald stopcriterium is bereikt. Enkele veel gebruikte stopcriteria zijn:

- De zoekopdracht is voltooid
- Een bepaalde waarde wordt bereikt (minimum aantal features of maximum aantal iteraties)
- Opeenvolgende toevoeging en verwijdering van features levert geen betere subset op
- Een voldoende goede subset is gegenereerd (bv. wanneer deze subset in staat is om een classificatieprobleem op te lossen met een bepaalde precisie)

Dan wordt de geselecteerde beste subset gevalideerd met voorkennis of via verschillende tests op andere datasets.

5.2 Classificatie

Een eerste techniek die toegepast wordt om een aanbevelingssysteem een beter beeld te geven van de documentencollectie, is classificatie. Door een artikel automatisch aan een categorie te kunnen toewijzen, krijgen we al een veel duidelijker beeld van waarover dit artikel nu precies handelt. Als we zowel de hoofd- als subcategorie van een artikel kunnen bepalen, hebben we al een eerste sterke indicatie of het artikel al dan niet interessant kan zijn voor de gebruiker en dus aanbevolen zal worden door het aanbevelingssysteem.

5.2.1 WEKA

Om zoveel mogelijk classificatiemethodes te kunnen testen wordt gebruik gemaakt van WEKA [17] (Waikato Environment for Knowledge Analysis). WEKA is een tool voor datamining ontwikkeld in de programmeertaal Java. Het bestaat uit een grafische werkomgeving voor het uitvoeren van de nodige stappen bij datamining, een CLI en is volledig ondersteund voor gebruik binnen een eigen Java-programma. Dit framework bevat in totaal 57 verschillende algoritmes om classificatie door te voeren. Deze zijn ingedeeld in verschillende categorieën: Bayes, functions, lazy, meta, mi, misc, rules en trees.

De versie van WEKA die gebruikt wordt in deze masterproef is 3.6.13.

De verschillende classifiers binnen WEKA ondersteunen geen meerdere niveaus van classificatie. Om toch gebruik te kunnen maken van een boomstructuur worden de artikels eerst ingedeeld in hun respectievelijke hoofdcategorieën en pas daarna in de subcategorieën. Omdat het mogelijk is dat voor deze gevallen een andere classifier een beter resultaat oplevert, gaan we deze apart testen. De preprocessing blijft voor alle gevallen gelijk. Wel gaan we eerst uitzoeken welke parameters van de preprocessing het beste beeld geven voor twee Bayesiaanse classifiers.

In eerste instantie wordt gefocust op de dataset van Het Laatste Nieuws. Vervolgens wordt met de gevonden inzichten en nieuwe classificatie toegepast op een tweede dataset van Wikipedia.

5.2.2 Preprocessing in WEKA

In WEKA wordt voor preprocessing van de documentencollectie de functie *string-to-wordvector* gebruikt. Deze functie converteert de teksten naar een set van attributen die het aandeel van dat attribuut voorstellen in de documentencollectie.

De string-to-word functie in WEKA neemt een aantal parameters. Een eerste parameter is de tokenizer functie. Deze bepaalt hoe een tekst wordt opgesplitst in woorden. Hier wordt gebruik gemaakt van een N-Gram tokenizer. Deze geeft de mogelijkheid om eventueel twee of drie woorden samen te nemen en als één woord in de documentencollectie te zien.

Ten tweede kan gekozen worden voor term frequency-inverse document frequency (tf-idf), al dan niet gecombineerd met een normalisatie. De achterliggende logica zit vervat in formule 5.1. Op deze manier worden termen die belangrijk zijn voor een document, maar minder vaak of niet in de volledige documentencollectie voorkomen, een groter gewicht toegekend in de vector representatie.

Een derde parameter die ingesteld kan worden is het gebruikte stemmingsalgoritme. Op basis van de metadata gegenereerd door Apache Tika (sectie 5.1.1) kan hier dynamisch een alternatieve stemmer gekozen worden per taal. Hier werd een Nederlandstalige stemmer gebruikt, zoals beschreven in [26] (aangezien Nederlandse datasets gebruikt worden). Deze stemmer is een uitbreiding van het originele stemmingsalgoritme van Porter, zoals voorgesteld in [36], naar de Nederlandse taal.

Een vierde parameter is de stopwoordfilter. Hiervoor wordt een lijst met stopwoorden gebruikt voor de Nederlandse taal. De stopwoordfilter zorgt ervoor dat woorden die veel voorkomen in een taal, maar relatief weinig inhoud hebben, niet overwogen worden om op te nemen in het woordenboek. Denk daarbij aan 'de', 'het', 'daar', 'dan', etc. Na de fase van tokenization kunnen deze woorden weggefilterd worden, zodat ze geen negatieve invloed hebben op het uiteindelijk resultaat.

Een vijfde parameter laat toe om de frequentie van een woord uit te drukken, in plaats van binaire aanwezigheid. Hierdoor wordt de belangrijkheid van bepaalde woorden soms meer in de verf gezet.

De laatste parameter is het aantal woorden dat behouden wordt per klasse uit de documentencollectie. Dit is natuurlijk een belangrijke parameter, omdat meer betekenisvolle woorden in het woordenboek naar een betere classificatie en/of clustering zullen leiden. Als we echter te veel woorden opnemen, waardoor er een bepaalde ruis gaat optreden, krijgen we ook slechts een suboptimale werking. Het is echter moeilijk om zomaar te bepalen hoeveel woorden een optimale classificatie zullen geven bij elke methode. Bovendien is het mogelijk dat het voor onze dataset niet nuttig is om een stemmer of stopwoordfilter te gaan gebruiken. Daarom wordt in sectie 5.2.3 een benchmark opgesteld die het optimale aantal woorden gaat onderzoeken bij

classificatie. Verder wordt daar ook het nut van de stemmer en van de stopwoordfilter verder onderzocht. Ook wordt het effect van deze parameters op de uiteindelijke classificatie geanalyseerd. In sectie 5.2.4 komen we terug op feature selection. Er wordt onderzocht of deze functie inderdaad een positief effect kan hebben op de effectieve classificatie.

5.2.3 Parameters voor preprocessing van data

Om de optimale parameters voor de preprocessing te bepalen wordt een classificatie uitgevoerd met de verwerkte dataset. We variëren de parameters en bepalen zo de optimale instellingen. De dataset die eerst gebruikt wordt is deze van Het Laatste Nieuws, januari 2008. De classifiers die gebruikt worden om de bekomen woordvector te testen zijn "Naive Bayes" en "Naive Bayes Multinomial".

Om een betrouwbaar resultaat te bekomen worden de tests met een *10-fold cross validation* [3] uitgevoerd. Dit is een techniek om voorspellende modellen te evalueren door de originele dataset te verdelen in een trainingsset om het model te trainen en een testset om deze te valideren. Bij 10-fold cross-validation wordt de originele dataset random verdeeld in 10 subsets. Daarvan wordt één subset behouden ter validatie van het model. De overige 9 subsets worden gebruikt als trainingsdata. Het cross-validation proces wordt 10 keer herhaald, waarbij elk van de subsets exact één keer gebruikt worden als validatieset. De 10 resultaten worden gecombineerd (uitgemiddeld) om een enkele voorspelling te maken. Het grote voordeel van deze methode is dat alle observaties zowel als trainings- en als validatiedata kunnen worden gebruikt.

Om de resultaten onafhankelijk te stellen van het aantal woorden die behouden worden van de documentencollectie, wordt gebruikt gemaakt van de standaard bijgehouden 1000 woorden. Deze parameter wordt in de volgende secties per dataset apart geverifieerd. We gaan er immers van uit dat er voor de classificatie op het eerste niveau van de boom meer woorden zullen nodig zijn om alle items correct te kunnen classificeren dan op het tweede niveau. Het lagere niveau bevat immers al tekst die specifiek is voor die categorie en heeft dus een minder uitgebreide woordenschat. Verder worden de verschillende parameters al dan niet ingesteld, zoals te zien in tabel 5.1. Hierbij staat "1" voor "gebruikt" en "0" voor "niet gebruikt".

Tabel 5.1: Gebruik van verschillende parameters bij preprocessing van de documentencollectie "Het Laatste Nieuws, januari 2008" en hun effect op het percentage correct geclassificeerde artikels volgens de "Naive Bayes" (NB) en "Naive Bayes Multinomial" (NBM) classifiers in hoofdcategorieën.

tf-idf	**Stemming**	**Stopwoorden**	**Frequentie**	**Tokenizer**	**NB**	**NBM**
0	0	0	0	1	69.85%	74.39%
0	0	0	0	2	67.62%	72.34%
0	0	0	0	3	67.44%	72.15%
0	0	0	1	1	56.06%	66.39%
0	0	0	1	2	54.26%	64.35%
0	0	0	1	3	54.24%	64.30%
0	0	1	0	1	71.50%	75.71%
0	0	1	0	2	69.26%	73.41%
0	0	1	0	3	68.91%	73.09%
0	0	1	1	1	58.01%	68.31%
0	0	1	1	2	56.25%	66.18%
0	0	1	1	3	56.20%	66.09%
0	1	0	0	1	70.85%	75.43%
0	1	0	0	2	68.19%	72.65%
0	1	0	0	3	68.35%	72.61%
0	1	0	1	1	57.68%	67.55%
0	1	0	1	2	55.74%	65.36%
0	1	0	1	3	55.88%	65.43%
0	1	1	0	1	**71.91%**	**76.37%**
0	1	1	0	2	69.44%	73.82%
0	1	1	0	3	69.42%	73.70%
0	1	1	1	1	59.09%	69.31%
0	1	1	1	2	57.49%	67.30%
0	1	1	1	3	57.29%	67.07%
1	0	0	0	1	36.40%	57.44%
1	0	0	0	2	40.78%	58.42%
1	0	0	0	3	40.85%	58.31%

1	0	0	1	1	40.88%	59.59%
1	0	0	1	2	45.99%	60.97%
1	0	0	1	3	45.68%	60.59%
1	0	1	0	1	34.45%	56.63%
1	0	1	0	2	39.73%	58.00%
1	0	1	0	3	39.73%	57.79%
1	0	1	1	1	38.56%	58.81%
1	0	1	1	2	43.64%	59.91%
1	0	1	1	3	43.97%	59.96%
1	1	0	0	1	38.92%	59.00%
1	1	0	0	2	42.52%	59.58%
1	1	0	0	3	42.72%	59.59%
1	1	0	1	1	43.99%	61.61%
1	1	0	1	2	47.16%	62.01%
1	1	0	1	3	47.29%	61.88%
1	1	1	0	1	37.17%	58.58%
1	1	1	0	2	41.38%	59.36%
1	1	1	0	3	41.70%	59.37%
1	1	1	1	1	41.78%	60.86%
1	1	1	1	2	45.68%	61.52%
1	1	1	1	3	46.10%	61.64%

De tabel leert ons dat een combinatie van een N-Gram tokenizer van maximum 1 woord, het stemmingsalgoritme en de stopwoordfiltering leidt tot het beste resultaat voor beide classificaties. Als we de frequentie weergeven in de woordvector, dan leidt dit meestal tot een daling van het aantal correct geclassificeerde documenten, terwijl de term frequency-inverse document frequency het aantal correct geclassificeerde documenten nog sterker doet dalen. Dit is een opmerkelijke vaststelling die niet lijkt te stroken met de verwachtingen. Mogelijke verklaringen voor dit fenomeen zijn dat de belangrijke termen in teksten van Het Laatste Nieuws niet veelvuldig gebruikt worden in artikels. Dit is bijvoorbeeld zo voor artikels waar de eerste minister vermeld wordt. De naam van de eerste minister wordt slechts één maal vermeld in het artikel en

vervolgens telkens vervangen door synoniemen en verwijzingen zoals “de eerste minister”, “de premier”, “Charles Michel”, etc. Het zou daarom kunnen dat andere woorden vaker voorkomen dan de eigenlijk typerende woorden voor dit artikel, waardoor een term frequency-inverse document frequency eigenlijk de belangrijkheid van deze termen onderdrukt. Een mogelijke oplossing voor dit probleem zou *synonym detection* (bv. [51], [49]) zijn. Verschillende andere oplossingen gaan met behulp van bestaande woordenboeken of encyclopedieën (bv. [7]) op zoek naar termen die gelijkaardig zijn. Deze technieken vallen echter buiten de scope van dit onderzoek en zijn bovendien nog weinig onderzocht voor Nederlandstalige teksten.

Om het vector space model op te stellen voor de dataset van Het Laatste Nieuws, gaan we dus een N-Gram tokenizer van 1 woord gebruiken en zowel stemming als stopwoordfiltering toepassen. Term frequency-inverse document frequency en frequentie van woorden gebruiken we niet.

Ook voor de dataset van Wikipedia worden de optimale parameters voor preprocessing bepaald op dezelfde manier. De resultaten zijn te vinden in tabel 5.2

Tabel 5.2: Gebruik van verschillende parameters bij preprocessing van de documentencollectie “Wikipedia” en hun effect op het percentage correct geclassificeerde artikels volgens de “Naive Bayes” (NB) en “Naive Bayes Multinomial” (NBM) classifiers in hoofdcategorieën.

tf-idf	**Stemming**	**Stopwoorden**	**Frequentie**	**Tokenizer**	**NB**	**NBM**
0	0	0	0	1	34.80%	37.56%
0	0	0	0	2	33.08%	35.53%
0	0	0	0	3	32.56%	35.04%
0	0	0	1	1	28.24%	33.54%
0	0	0	1	2	28.46%	32.89%
0	0	0	1	3	28.26%	32.63%
0	0	1	0	1	36.09%	39.40%
0	0	1	0	2	33.86%	36.71%
0	0	1	0	3	33.69%	36.23%
0	0	1	1	1	30.36%	35.24%
0	0	1	1	2	29.16%	33.62%
0	0	1	1	3	28.42%	33.17%
0	1	0	0	1	35.53%	38.35%

0	1	0	0	2	33.79%	36.41%
0	1	0	0	3	33.41%	35.83%
0	1	0	1	1	27.97%	34.04%
0	1	0	1	2	28.56%	33.42%
0	1	0	1	3	28.46%	33.30%
0	1	1	0	1	36.23%	39.65%
0	1	1	0	2	34.61%	37.81%
0	1	1	0	3	34.08%	37.12%
0	1	1	1	1	29.86%	35.49%
0	1	1	1	2	29.59%	34.69%
0	1	1	1	3	29.19%	34.49%
1	0	0	0	1	37.03%	40.22%
1	0	0	0	2	35.29%	38.33%
1	0	0	0	3	34.36%	37.67%
1	0	0	1	1	37.97%	41.06%
1	0	0	1	2	37.00%	39.78%
1	0	0	1	3	35.53%	38.86%
1	0	1	0	1	37.38%	40.54%
1	0	1	0	2	35.57%	38.57%
1	0	1	0	3	34.52%	37.84%
1	0	1	1	1	38.32%	41.46%
1	0	1	1	2	36.69%	39.77%
1	0	1	1	3	35.99%	39.18%
1	1	0	0	1	37.89%	40.91%
1	1	0	0	2	36.11%	39.53%
1	1	0	0	3	35.14%	38.70%
1	1	0	1	1	38.82%	41.82%
1	1	0	1	2	37.47%	40.65%
1	1	0	1	3	36.11%	39.74%
1	1	1	0	1	37.87%	40.99%
1	1	1	0	2	36.51%	39.95%
1	1	1	0	3	35.24%	38.89%

1	1	1	1	1	**39.27**%	**42.28**%
1	1	1	1	2	37.64%	40.94%
1	1	1	1	3	36.26%	40.01%

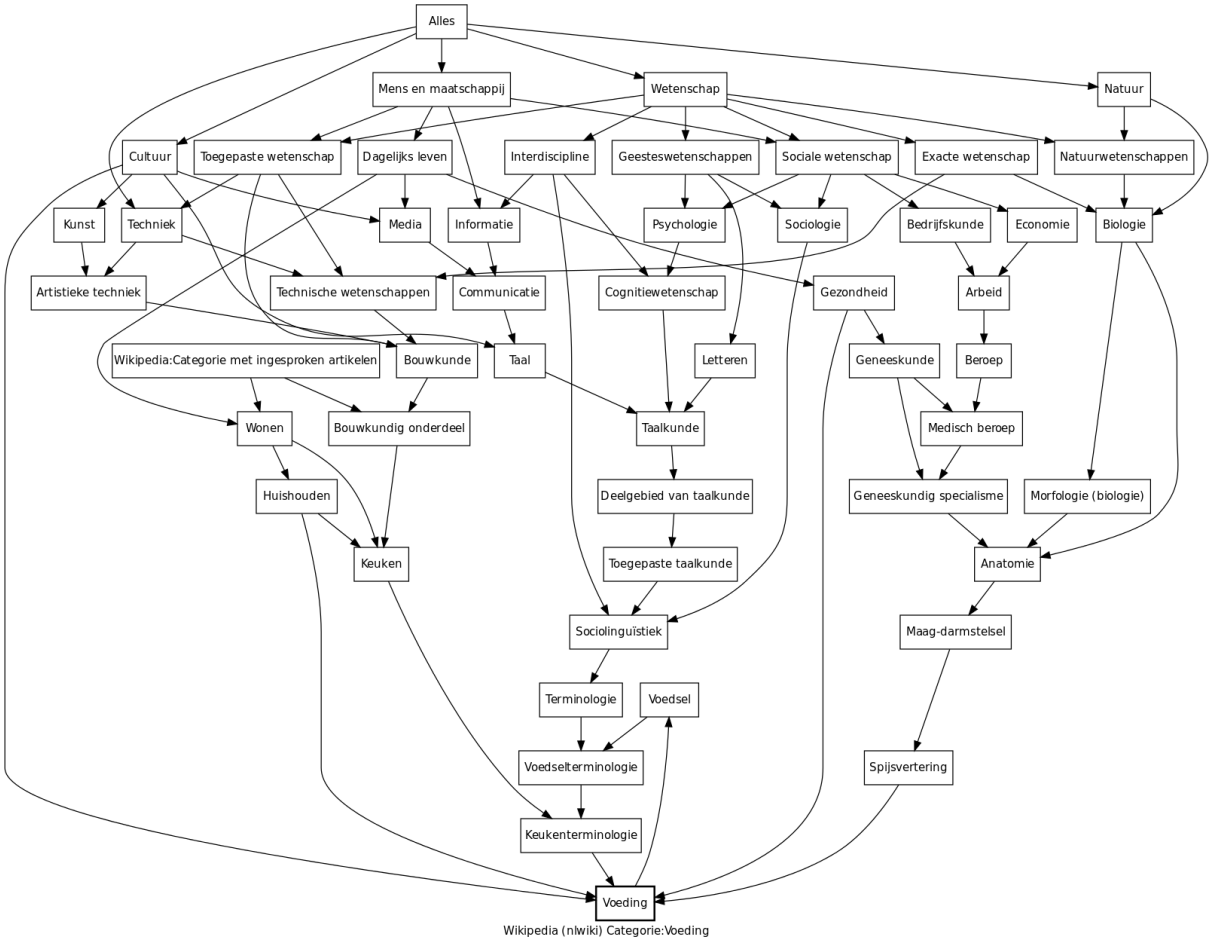
Er kan opgemerkt worden dat over hele lijn, onafhankelijk van de ingestelde parameters, het aantal correct geclassificeerde artikels een stuk lager ligt dan bij Het Laatste Nieuws het geval was. Een mogelijke verklaring ligt in het feit dat de dataset van Wikipedia zo is opgebouwd dat deze per categorie evenveel artikels bevat. Als een classifier dus niet zeker is waar een bepaald artikel terecht moet komen en bijgevolg gaat gokken, heeft het systeem maar een kans van 1 op 9 om het artikel alsnog in een juiste categorie onder te brengen. Als we dit vergelijken met Het Laatste Nieuws zien we daar 11 categorieën. De verdelingen van de artikels over die categorieën is echter niet evenredig. Als een classifier bij deze dataset niet zeker is waar een artikel thuishoort, dan gokt hij op de categorie waar de meeste artikels inzitten. Voor de datset van januari 2008 bevat de categorie nieuws 31,69% van alle artikels, wat dus een kans van bijna 1 op 3 vertegenwoordigd. Het is dus logisch dat een classifier bij de dataset van Wikipedia een beduidend lagere score haalt voor het aantal correct geclassificeerde artikels.

Een andere verklaring voor dit fenomeen is dat de categorieën waarin de artikels van Wikipedia werden ingedeeld veel minder afgebakend zijn dan bij Het Laatste Nieuws. In principe is de structuur van Wikipedia opgebouwd als een hiërarchie, weergegeven volgens een acyclische verbonden graaf [32]. Deze structuur lijkt op een boomstructuur, maar iedere knoop kan verbonden zijn met verschillende andere knopen rondom zich. Er zitten bijgevolg veel overlappingen tussen de categorieën en er komen zelfs circulaire referenties terug. Als we bijvoorbeeld een categorie "cultuur" beschouwen vinden we daar artikels over o.a. voeding. Mochten we de graaf voor categorie voeding opstellen krijgen we ongeveer de figuur in 5.2[2].

Deze figuur toont hoe de categorie voeding niet enkel onder hoofdcategorie cultuur staat, maar ook onder hoofdcategorieën "mens en maatschappij", "wetenschap", "natuur", etc. Het is bijgevolg al een goed resultaat dat bepaalde artikels in wezen ingedeeld worden in een correcte structuur. De resultaten van deze dataset vallen dan ook moeilijk te vergelijken met deze van Het Laatste Nieuws, waar wel een duidelijke boomstructuur van 2 niveaus wordt gebruikt. De

[2] `https://tools.wmflabs.org/vcat/catgraphRedirect?wiki=nlwiki&cat=Voeding&format=png&links=wiki`

Figuur 5.2: Hiërarchie binnen Wikipedia van de categorie voeding



verdere resultaten van classificatie van deze artikels kunnen gevonden worden in sectie 5.2.8.

Als vervolgens gefocust wordt op de preprocessing parameters, valt op dat hier term frequency-inverse document frequency wel een positief resultaat oplevert. De optimale preprocessing parameters voor deze dataset liggen meer in de lijn van verwachting dan deze van het laatste nieuws, aangezien alle mogelijkheden van de preprocessing bijdragen tot een beter resultaat van classificatie. Er wordt bijgevolgd bij preprocessing van deze dataset zowel tf-idf, stemming, stopwoordfiltering, woordfrequentie en tokenization met 1 NGram gebruikt om tot een optimale classificatie te kunnen komen.

5.2.4 Feature selection

Feature selection probeert om een subset van de features te zoeken die relevant zijn voor het doelconcept. Het wordt op verschillende manieren beschreven, maar voor deze use case zijn de

criteria de volgende:

- De nauwkeurigheid van de classificatie daalt niet of nauwelijks,
- De resulterende klassenverdeling ligt zo dicht mogelijk bij de originele klassenverdeling, gegeven alle features.

Om aan deze voorwaarden te voldoen gebruiken we het Information Gain (IG) algoritme [21]. Dit algoritme werkt als volgt: stel dat P_i de globale probabiliteit is van klasse i voorstelt, en $p_i(w)$ de probabiliteit van klasse i als het document het woord w bevat. $F(w)$ is dan de fractie van de documenten die het woord w bevatten. Information Gain $I(w)$ is als volgt gedefinieerd voor een gegeven woord w:

$$I(w) = -\sum_{i=1}^{k} P_i \cdot \log(P_i) + F(w) \cdot \log(p_i(w)) + (1 - F(w)) \cdot \sum_{i=1}^{k} (1 - p_i(w)) \cdot \log(1 - p_i(w)) \quad (5.2)$$

Hoe groter de waardes voor IG $I(w)$, hoe groter de discriminerende kracht van het woord w [13].

We testen de implementatie van IG uit in combinatie met het Ranker-algoritme in WEKA, op het woordenboek met de gekozen opties bij preprocessing voor de dataset van Het Laatste Nieuws. Dit levert de resultaten in tabel 5.3.

Tabel 5.3: Vergelijking voor en na feature selection met de Information Gain-methode voor dataset Het Laatste Nieuws, januari 2008

Origineel			**Information Gain**		
NB	**NBM**	**woorden**	**NB**	**NBM**	**woorden**
71.91%	76.37%	1000	72.01%	76.51%	893

We merken een sterke daling in het aantal woorden (10,70%), maar tergelijkertijd een lichte stijging in het aantal juist geclassificeerde documenten (voor de "Naive Bayes" classifier 0,10%, voor de "Naive Bayes Multinomial"classifier 0,14%). We kunnen hier dus besluiten dat IG een algoritme is dat goede resultaten geeft voor deze tekstuele data. De zogenaamde ruis wordt weggenomen uit de woordvector en de belangrijkste woorden werden behouden.

Als we diezelfde IG toepassen op de dataset van Wikipedia krijgen we de resultaten in tabel 5.4.

Bij deze dataset merken we een heel sterke daling in het aantal woorden op (40 %). Dit duidt op het feit dat veel woorden gevonden worden die weinig bijdragen tot een categorie. Ook dit

Tabel 5.4: Vergelijking voor en na feature selection met de Information Gain-methode voor dataset Wikipedia

Origineel			**Information Gain**		
NB	**NBM**	**woorden**	**NB**	**NBM**	**woorden**
39.26%	42.27%	1000	34.93%	44.92%	600

kan herleid worden naar het feit dat de classifier er moeite mee heeft een eenduidige definitie op te stellen van een categorie. Ondanks de sterke daling in woorden zien we een stijging in het aantal correct geclassificeerde artikels voor de NBM-methode (2,65%). Voor de naïeve Bayesiaanse methode merken we wel een daling op van 4,33 %. Ondanks deze daling bewijst het IG algoritme zijn nut door het aantal woorden sterk te doen dalen en tegelijkertijd een stijging in het aantal correct geclassificeerde artikels te bekomen voor de NBM-methode.

5.2.5 Classificatie in hoofdcategorieën (dataset Het Laatste Nieuws)

Er wordt gestart met het classificeren in hoofdcategorieën. De dataset van januari 2008 bevat 11 hoofdcategorieën: Auto, Bizar, Geld, iHLN, Muziek, Nieuws, Reizen, Showbizz, Sport, Wetenschap en You. De verdeling van de artikels over deze hoofdcategorieën kan gevonden worden in tabel 3.1.

Om de best mogelijke classificatie te vinden, wordt een testbank opgesteld waarin een selectie van algoritmes voor classificatie in WEKA (46) opgenomen zijn. Enkele classifiers werden niet opgenomen omdat ze o.a. manuele input verwachten. Omdat dit een geautomatiseerde toepassing is kunnen deze algoritmes niet manueel verwerkt worden. De gekozen classifiers worden met de standaardopties uitgevoerd op de data van Het Laatste nieuws uit januari 2008, die vooraf door de preprocessing-fase gegaan is. De classifiers worden vervolgens gerangschikt volgens het percentage correct geclassificeerde artikels, gebruik makend van een 10-fold cross-validation (zie 5.2.3).

Naïeve baseline classifier

Het aantal woorden die opgenomen wordt in de preprocessing van de documentencollectie wordt gevarieerd van 100 t.e.m. 20000 attributen met telkens een increment van 100 attributen. Nadien wordt de Information Gain methode (zie 5.2.4) toegepast om eventuele ruis te verwijderen. Snel

- na 2000 attributen - merken we echter dat enkele methodes het heel erg slecht doen. Een eerste zeer naïeve classifier waarmee vergeleken wordt is een classifier die alle artikels toekent aan de categorie met het meeste artikels (in dit geval nieuws). Deze naïeve implementatie zal in principe 31,69% correct classificeren. In WEKA wordt deze geïmplementeerd onder de naam ZeroR. Dit algoritme is ook opgenomen in lijst van classifiers.

Als we ZeroR als baseline classifier beschouwen om te bepalen wat een slechte classifier is, dan vinden we enkele methodes die deze baseline niet halen en die het m.a.w. slechter of net even goed doen als ZeroR bij 2000 woorden. De zwakke classifiers zijn VFI (Voting Feature Intervals), Vote, Stacking, StackingC, MultiScheme, Grading, CVParameterSelection en ClassificationViaClustering. Als we enkele van deze algoritmes van dichterbij gaan bekijken, zien we dat o.a. Vote, VFI, Stacking, StackingC, MultiScheme en Grading enkele extra parameters verwachten waarin (enkele) andere classifiers worden opgegeven. Die methodes gaan dus andere classifiers combineren en enkelen hiervan worden verder besproken in 5.2.6, waar de voordelen van het combineren van classifiers wordt onderzocht.

Het is voor het verder onderzoek naar de beste classifiers voor deze dataset niet nuttig dat de vermelde classifiers verder individueel onderzocht worden. Om rekenkracht te sparen worden deze dan ook niet meer opgenomen in de resultaten na 2000 woorden.

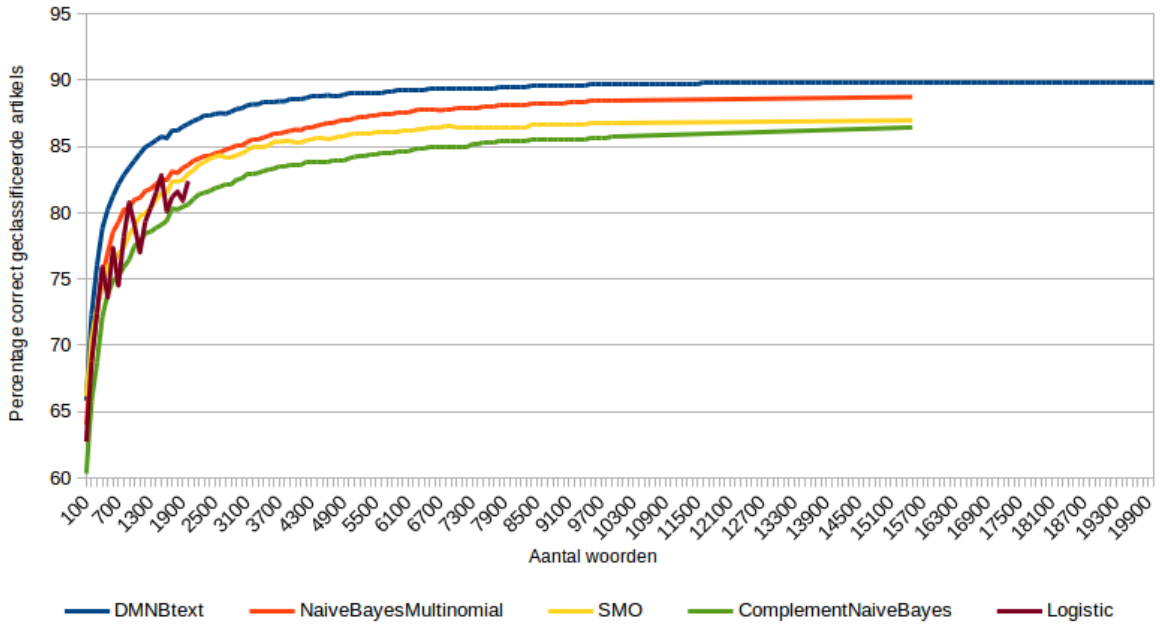
Top 10 classifiers

Na 2000 woorden wordt bovendien een top 10 van beste classifiers opgesteld. Criteria voor selectie zijn methodes die steeds verbeteren als het aantal woorden wordt opgedreven en methodes die binnen maximum 2 uur een oplossing kunnen vinden. Op die manier kan sneller en gemakkelijker vergeleken worden. Grafiek 5.3 toont vijf beste classifiers en het punt waar hun optimale classificatie bereikt wordt. Het is duidelijk dat de classifier DMNBText hier het best presteert vanaf 15500 woorden (89,86% correct geclassificeerd). De classifier Logistic heeft vanaf 2000 woorden geen geheugen meer genoeg om verdere classificatie toe te staan, maar presteert wel goed genoeg bij weinig woorden om in de top 10 te worden opgenomen.

5.2.6 Combineren van classifiers voor hoofdcategorieën

Het is mogelijk dat we door verschillende goede individuele classifiers te combineren, een betere algemene classificatie bekomen. Om die stelling te onderzoeken worden drie combinatiealgorit-

Figuur 5.3: Top 5 classifiers voor classificatie van dataset Het Laatste Nieuws in hoofdcategorieën



mes getest in de volgende paragrafen: voting, stacking en grading.

Voting

Het algoritme dat eerst gebruikt wordt is “Vote” [24]. Dit is een algoritme dat zelfstandig geen classificatie uitvoert (zie ook resultaten individuele classifiers, 5.2.5). Het neemt een aantal zelf gekozen classifiers samen en laat die stemmen om te bepalen tot welke klasse een bepaalt document behoort. Om er zeker van te zijn of dit een effectieve verbetering oplevert passen we het ”Vote”-algoritme toe met de top 4 beste algoritmes voor classificatie in hoofdcategorieën: DMNBText, MultinomialNaiveBayes, SMO en ComplementNaiveBayes.

Het Vote-algoritme laat toe om met verschillende methodes de originele classificaties te combineren. Deze worden voorgesteld in tabel 5.5, samen met het percentage correct geclassificeerde artikels voor de preprocessed dataset van januari 2008.

De methode “Majority Voting” geeft een licht beter resultaat dan de beste individuele classifier (DMNBText; 89,86% correct geclassificeerd). Het verschil is dan wel heel klein (0,06%), maar toch significant aangezien deze classifier op een dataset van bijvoorbeeld 10000 artikels toch 600 artikels meer correct zou kunnen classificeren.

Tabel 5.5: Aantal correct geclassificeerde artikels bij gebruik van het Vote-algoritme met classifiers DMNBText, MultinomialNaiveBayes, SMO en ComplementNaiveBayes voor dataset Het Laatste Nieuws (januari 2008) met verschillende combinatiemethodes.

Combinatiemethode	Correct geclassificeerd
Average of Probabilities	89,65%
Product of Probabilities	86,45%
Majority Voting	89,92%
Minimum Probability	86,45%
Maximum Probability	86,56%

Stacking

Een andere methode om mogelijk een beter classificatie te bekomen is het “Stacking”-algoritme. Meestal presteert dit beter dan voting [45]. Stacking wordt gezien als een generalisatie van het ”Vote”-algoritme. Het combineert de voorspellingen van verschillende andere classifiers en past een zelfgekozen combinatie-algoritme toe met de voorspellingen van de andere algoritmes als extra input.

Het combineert meerdere classifiers die gegenereerd zijn door verschillende lerende algoritmes L_1,...,L_n op een dataset S toe te passen. De dataset bestaat uit elementen in de vorm van $s_i = (x_i, y_i)$, welke in dit geval de paren van de feature vectoren (x_i) met hun classificatie (y_i) voorstellen. In de eerste fase worden base-level classifiers C_1,...,C_n gegenereerd, waar $C_i = L_i(S)$. In de tweede fase wordt een meta-level classifier geleerd om de output van de base-level classifiers te combineren.

Ook hier passen we stacking toe met een lineaire regressie methode en de vier beste individuele classifiers. Dit levert een verbetering op tot 89,98%. Dit is nog iets beter (0,06%) dan wat bij voting kon bereikt worden.

Grading

Een andere methode om mogelijk een beter classificatie te bekomen is met het ”Grading”-algoritme [43]. *Grading* maakt gebruikt van *graded* voorspellingen voor een trainingsset van meta classifiers die leren voorspellen wanneer de basisclassifier correct is. De trainingsset van

deze meta classifiers is geconstrueerd gebruik makend van graded voorspellingen van de corresponderende basisclassifier als de nieuwe klasselabels voor de originele attributen.

Door gebruik te maken van het grading algoritme met de top 4 beste classifiers wordt 89,70% juist geclassificeerd. Deze methode presteert dus minder goed dan de vorige twee combinatiealgoritmes voor dit classificatieprobleem, maar iets beter dan de beste individuele classifier.

Conclusie

Door gebruik te maken van combinatiealgoritmes kan voor deze dataset 0,12% beter geclassificeerd worden in hoofdcategorieën. De beste classfier was het grading-algoritme, gevolgd door voting en stacking. Wel kon met elk van deze methodes een betere classificatie behaald worden dan met de beste individuele classifier.

5.2.7 Classificatie in subcategorieën (dataset Het Laatste Nieuws)

Het is mogelijk dat een classificatie in subcategorieën een ander algoritme vereist dan het classificeren in hoofdcategorieën. Daarom wordt ook voor de subcategorieën getest welke classifier de beste resultaten geeft. Voor subcategorieën wordt gezocht naar de beste individuele uit de top 10 of de beste gecombineerde classifier uit de top 4 beste algoritmes (DMNBtext, SMO, NaiveBayesMultinomial, ComplementNaiveBayes). De belangrijkste resultaten zijn te vinden in tabel 5.6.

De veronderstelling dat voor elke subklasse een andere classifier gebruikt moet worden met een ander aantal woorden klopt. Als bijvoorbeeld de categorie “you” vergeleken wordt met “muziek”, dan zit een groot verschil tussen enerzijds het aantal woorden die nodig zijn (resp. 13000 en 1000) en anderzijds het aantal juist geclassificeerde artikels. De grootste oorzaak hiervan is dat het classificatieprobleem dat gecreëerd werd voor de categorie “You” veel moeilijker is dan dat voor de categorie “muziek”. Zo zijn er veel meer artikels binnen die categorie en zijn er bovendien veel meer subcategorieën. Dit zorgt ervoor dat de classifier een moeilijkere beslissing moet nemen en bijgevolg meer artikels verkeerd klasseert tegenover een gemakkelijker probleem.

5.2.8 Classificatie in hoofdcategorieën (dataset Wikipedia)

Zoals eerder aangehaald in sectie 5.2.3 is de dataset van Wikipedia veel moeilijker te classificeren. De top 10 classifiers werden ook hier toegepast met een variabel aantal woorden.

Tabel 5.6: De best scorende classifiers voor elke subcategorie van dataset Het Laatste Nieuws.

	woorden	classifier	percentage
nieuws	9000	DMNBtext	93.77%
sport	7000	Vote	93.77%
ihln	1000	Stacking	92.00%
muziek	1000	Grading	100.00%
showbizz	4000	Stacking	84.00%
you	13000	Stacking	78.98%
geld	8000	Stacking	96.41%

5.2.9 Classificatie in subcategorieën (dataset Wikipedia)

5.3 Clustering

Clustering is een methode die gebruikt wordt om documenten met gelijkaardige inhoud in eenzelfde cluster te groeperen [20]. Het wordt naast classificatie toegepast om gelijkaardige documenten te vinden over de grenzen van de classificatie heen (documenten die aan verschillende categorieën toebehoren, maar bv. over dezelfde persoon handelen). Ook zal clustering ervoor zorgen dat documenten binnen dezelfde categorie toch nog verder opgedeeld kunnen worden in bepaalde topics. Op deze manier kan meer informatie uit de teksten gehaald worden en die informatie kan op zijn beurt gebruikt worden door het aanbevelingssysteem.

Elke cluster bevat een aantal documenten. Het is belangrijk dat documenten in een bepaalde cluster gelijkaardig zijn aan elkaar, maar verschillen met documenten in andere clusters. Cluster methodes groeperen documenten op basis van hun distributie in een bepaalde documentenruimte. We krijgen zo bijvoorbeeld een n-dimensionele ruimte als we het vector space model zouden gebruiken.

Clustering wordt berekend op basis van attributen van de tekst. Nochtans is het concept "cluster" een subjectief begrip. Wat de ene persoon ziet als een goede en relevante cluster, kan door iemand anders niet als een goede cluster aanzien worden. Clustering vraagt bijgevolg om een duidelijk evaluatiecriterium (zie 5.3.2), zodat we objectief kunnen vaststellen wat een goede cluster is.

5.3.1 K-Means Clustering algoritme

Een populair clustering algoritme is het K-Means clustering algoritme (ook wel het algoritme van Lloyd genoemd [29]). Voor een gegeven dataset $X = x_1, ...x_n, x_n \in R^d$ wordt geprobeerd om deze data in M verschillende subsets (clusters) $C_1, ..., C_M$ te partitioneren, zodat een bepaalde clustervoorwaarde voldaan is [27]. Het algoritme verwacht daarvoor een bepaalde waarde voor het aantal clusters k die moeten gevormd worden. Deze k zorgt voor heel wat belemmeringen bij het uitvoeren van dit algoritme, vooral omdat het vooraf niet duidelijk is hoe groot clusters zullen of kunnen worden en omdat dit voor elke documentencollectie anders ligt. Als een k te klein wordt genomen zullen artikels die ver van elkaar liggen toch samen geclusterd worden. Als we de k te groot nemen zullen de onderwerpen misschien onnodig opgesplitst worden in (te) kleine clusters.

Daarom wordt in eerste instantie het K-means algoritme iteratief uitgevoerd voor elke waarde van k. Het nadeel van elke k iteratief uit te proberen is uiteraard de uitvoeringstijd bij grote documentencollecties. Om deze te beperken worden een aantal verschillende methodes uitgeprobeerd met als doel de uitvoeringstijd te verlagen (zie 5.3.4, 5.3.5).

Een tweede probleem bij het k-means clustering algoritme is dat het er steeds opnieuw op zoek moet gaan naar de optimale clustercentra. Daartoe moet een aantal iteraties ingesteld worden waarin het algoritme vervolgens op zoek gaat naar de ideale centra. De clusters worden in eerste instantie random gekozen en proberen vervolgens de inter-clusterafstand (tussen verschillende clusters) te vergroten en de intra-clusterafstand (tussen punten binnen dezelfde cluster) te verkleinen. Omdat dit een proces is dat veel tijd in beslag neem wordt in 5.3.5 een oplossing voorgesteld die dit probeert te minimaliseren.

5.3.2 Evaluatiecriteria voor clustering

Er bestaan twee manieren om resultaten van clustering te evalueren. Enerzijds kunnen statistische functies gebruikt worden om de eigenschappen van de clustering te beschrijven. Anderzijds kan classificatie gebruikt worden als “gold standard”, waarbij de clustering vergeleken wordt met de gegeven classificatie. Omdat het doel van de clustering in deze toepassing echter niet is om classificatie door te voeren, zijn we aangewezen op de statistische functies. Om te bepalen wat een goede cluster is, moeten we voor het k-means algoritme bepalen hoeveel clusters k een dataset bevat. Hiervoor worden enkele criteria besproken.

Vuistregel

De vuistregel [25] is een simpele regel, die toegepast kan worden op elke gegeven dataset. De uitkomst van deze regel wordt enkel als een indicatie beschouwd en kan zeker niet gebruikt worden voor het bepalen van het effectieve aantal clusters.

$$k = \sqrt{\frac{n}{2}} \tag{5.3}$$

Elbow method

De "elbow method" [25] is een visuele methode. Het idee is om alle waarden van k te itereren en telkens een kostenfunctie te plotten op een grafiek. Bij een bepaalde waarde van k daalt de kost sterk, waarna het een plateau bereikt als k verder geïtereerd wordt.

Het is echter niet altijd gemakkelijk om deze "elbow" te vinden, zeker niet indien de documentencollectie groot wordt. Ze vereist enerzijds dat we alle clusters aflopen, maar vereist anderzijds ook dat een functie kan gemaakt worden die de optimale k kan vinden. Die functie werd gevonden in [48] met de zogenaamde "Gap statistic". Deze methode is echter niet bruikbaar voor grote documentencollecties omdat ze vereist dat eerst alle clusters worden gevonden, voor de optimale k kan bepaald worden. Voor een grote documentenset is deze methode dus niet bruikbaar.

Inter- versus intra-cluster afstand

Dit criterium werd voorgesteld in O.a. [38] en probeert om de som van de kwadratische afstanden van alle punten tot hun cluster centrum te minimaliseren. Hiervoor moeten we de afstanden van elk document tot cluster centrum berekenen. Deze worden gegeven door:

$$intra = \frac{1}{N} \sum_{i=1}^{K} \sum_{x \in C_i} ||\mathbf{x} - \mathbf{z_i}||^2 \tag{5.4}$$

waarbij N het aantal documenten is in de documentencollectie, K het aantal clusters en z_i het cluster centrum van cluster C_i.

Vervolgens berekenen we de inter-cluster afstand. We berekenen dit als het minimum van de afstanden tussen de cluster centra:

$$intra = min(||\mathbf{z_i} - \mathbf{z_j}||^2); i = 1, 2, ..., K-1; j = i+1, ..., K \tag{5.5}$$

Van deze reeks hebben we enkel het minimum nodig, omdat we deze waarde willen maximaliseren. De andere clusters liggen sowieso verder van elkaar.

We combineren beide waarden vervolgens tot de "validity measure":

$$validity = \frac{intra}{inter} \tag{5.6}$$

Omdat de intra-cluster afstand steeds geminimaliseerd moet worden en de inter-cluster afstand steeds gemaximaliseerd, moet de validity measure telkens geminimaliseerd worden (minimum van de teller delen door maximum in de noemer zorgt voor minimalisatie van de validity measure). Omdat deze validity measure steeds daalt wordt een bepaalde *treshold* ingesteld, zodat de clusters niet te specifiek worden (en dus heel weinig relevante buren zouden overhouden). Deze waarde wordt experimenteel bepaald in tabel 5.7 voor een kleine dataset (20 documenten). Daarbij wordt het iteratieve k-means clustering algoritme toegepast op de dataset en bij elke uitvoer de validity-measure nakijken. De teksten zijn gekozen uit de dataset van Het Laatste Nieuws en zijn zo geselecteerd dat er artikels van 5 totaal verschillende topics aanwezig zijn (de zwangerschap van Angelina Jolie, Osama Bin Laden, transfers van voetbalploeg Anderlecht, Maddie McCan en de Amerikaanse presidentsverkiezingen).

Het clusteralgoritme dat gebruikt wordt past k-means clustering toe met een feature vector van 1000 features. Het algoritme krijgt 100 iteraties om de optimale cluster centra te bepalen voor k-means en dit voor elke k. Iedere iteratie wordt a.d.h.v. de vooraf opgesteld lijst van topics vergeleken hoe de clustering presteert. Naast de Euclidean distance wordt nog een andere afstandsfunctie [22] gebruikt: Cosine distance. Deze zou beter presteren voor tekstuele data. Bij de Cosine distance [35] worden punten voorgesteld als vectoren van de oorsprong naar het punt in de ruimte. Twee vectoren vormen op die manier een bepaalde hoek θ. Wanneer die hoek klein is liggen de punten enigszins dicht bij elkaar. De cosinus van die hoek ligt dicht bij 1 wanneer de hoek klein is en wordt kleiner wanneer de hoek groter wordt. De Cosine distance functie trekt de waarde van de cosinus af van 1 om een afstand te bekomen. Deze is dus 0 indien de punten dicht van elkaar liggen en wordt groter naar mate de punten verder uit elkaar liggen. De formule voor twee n-dimensionele vectoren $(a_1, a_2, ..., a_n)$ en $(b_1, b_2, ..., b_n)$ is:

$$d = 1 - \frac{(a_1b_1 + a_2b_2 + ... + a_nb_n)}{\sqrt{a_1^2 + a_2^2 + ... + a_n^2}\sqrt{b_1^2 + b_2^2 + ... + b_n^2}} \tag{5.7}$$

Tabel 5.7: Berekenen van de validity-measure voor verschillende afstandsfuncties (euclidean distance, Cosine distance) bij k-means algoritme (1000 features, 100 iterations)

k	euclidean	cosine
1	1.80	1.80
2	0.48	1.13
3	0.20	0.63
4	0.17	0.61
5	0.17	0.51
6	0.17	0.42
7	**0.16**	0.37
8	0.12	**0.33**
9	0.12	0.29
10	0.12	0.21
11	0.10	0.22
12	0.09	0.19
13	0.07	0.15
14	0.05	0.13
15	0.04	0.09

De subjectief bepaalde optimale validity measures zijn in het vet aangeduid. Dit zijn de validity measures waarbij de topics het best lijken op de vooropgestelde topics (of delen daarvan). Zoals verwacht komt de Cosine distance het dichtst in de buurt van de effectieve topics. Geen enkel algoritme slaagt er echter in de vooropgestelde 5 topics te detecteren. Omdat de cosine distance het best presteert wordt verder gewerkt met deze afstandsfunctie.

5.3.3 Named entity recognition

Veel van de clusters die geconstrueerd worden handelen over een bepaald onderwerp. In de recente actualiteit zouden we bijvoorbeeld een cluster rond ISIS kunnen vinden die handelt over terrorisme. Die bepaalde cluster zou geïdentificeerd kunnen worden door zogenaamde *named entities* van personen binnen deze organisatie, personen die aanslagen pleegden, de plaatsen waar aanslagen plaatsvonden, etc.

Dit idee werd geïntroduceerd door o.a. [33], waar een systeem gebouwd werd dat artikels clusterde uit twee verschillende talen door gebruik te maken van *natural language processing* (NLP) en named entity recognition. Zij gaan uit van het idee dat een artikel (en bij uitbreiding een tekst) geschreven is met zes vragen in het achterhoofd: wat, wie, wanneer, waar, waarom en hoe. Een belangrijk deel van deze vragen kan beantwoord worden met een named entity (NE). Antwoorden op *waar* geven bijvoorbeeld namen van locaties, terwijl wanneer een datum of tijd teruggeeft. Als ervan uit gegaan wordt dat:

- NEs over verschillende, gelijkaardige documenten behouden blijven omdat het moeilijk is om een naam te parafraseren,
- de frequentie van NEs in zinnen de belangrijkheid benadrukt van gebeurtenissen waar ze mee geassocieerd zijn,
- twee verschillende verhalen met dezelfde NEs meer dan waarschijnlijk aan dezelfde topic toebehoren,

dan kan een model gemaakt worden dat specifieke clusters rond deze NEs kan bouwen.

Omdat het gemakkelijker ligt om topics te vinden rond personen (PERS), organisaties (ORG), locaties (LOC) en andere (MISC) worden enkel deze NEs gebruikt voor het model. Andere categorieën zoals tijd, datum of cijfers worden niet als belangrijk gezien omdat ze een hoge kans

geven op extra ruis als ze niet voorzichtig gebruikt worden. Er kunnen bijvoorbeeld twee belangrijke gebeurtenissen in de geschiedenis gebeurd zijn op hetzelfde tijdstip, maar de gebeurtenissen kunnen totaal verschillen. Zeker bij datasets zoals die van Het Laatste Nieuws, waar veel topics op dezelfde dag worden behandeld, moet daarvoor opgelet worden.

Sommige auteurs veronderstellen echter dat documenten clusteren enkel rond NEs schadelijk kan zijn voor clustering (o.a. [14], [15]). Ze veronderstellen dat NEs alleen niet genoeg informatie geven over een tekst om een dataset correct te clusteren. EEn voorbeeld hiervan is een tekst over tuinieren, waarin aangegeven wordt dat hagen moeten gesnoeid worden rond deze periode van het jaar, welke planten gezaaid kunnen worden, etc. Deze tekst bevat geen NEs en kan dus niet geclusterd worden. Daarom wordt in dit geval een model gebruikt waarin NEs een belangrijkere rol krijgen dan andere features. Nadat de documentencollectie het stadium van preprocessing doorlopen heeft en de feature vector wordt opgebouwd, krijgen NEs een groter gewicht toegekend dan de andere features.

Implementatie van NLP

Om NEs uit een tekst te halen wordt gebruik gemaakt van de Stanford Named Entity Recognizer [12] met een Nederlandse classifier. Binnen dit systeem worden NEs getagd als volgt: PER (personen), ORG (organisaties), LOC (locaties) en MISC (andere). Door de term frequency van deze NEs te verhogen worden ze belangrijker voor een tekst en wordt geprobeerd om een betere clustering te verkrijgen.

Omdat deze classifier enkel voor het Nederlands werkt, zijn er voor deze classifier geen grote evaluatiesets beschikbaar. Daarom wordt een set van 1930 woorden zelf getagd waarop vervolgens geëvalueerd wordt. De globale precisie van deze classifier was van 97,31% (1878/1930 objecten werden correct getagd). De individuele scores kunnen gevonden worden in tabel 5.8.

De NLP wordt geïntegreerd tijdens de preprocessing. Tussen de stap van tokenization en stopwoordfiltering komt een extra stap die de NEs uit een tekst een hogere frequentie geeft tegenover de andere elementen. Op die manier wordt een woord belangrijker voor een tekst en wordt verwacht dat hij vaker als discriminerende factor zal dienen bij het clusteren.

Tabel 5.8: Precisie van een Nederlandse classifier voor Stanford NLP

	Aantal tags	antwoorden	Precisie	Recall	F-score
LOC	26	22	0.95	0.81	0.88
PER	30	28	0.75	0.7	0.72
MISC	22	18	0.67	0.55	0.6
ORG	7	17	0.35	0.85	0.5
O	1845	1793	0.97		

Resultaten voor kleine datasets

Er wordt terug gegrepen naar de dataset met 20 documenten en 5 topics uit de vorige sectie. Opnieuw wordt dezelfde clustering toegepast, maar krijgen de named entities deze keer een groter gewicht. De volgende clustering an gevonden worden met een validity measure van 0.31:

Cluster 0

Document: 0, Title: Pakistan niet op zoek naar bin Laden

Document: 5, Title: Hoe Osama bin Laden de Amerikanen ontglipte in 2001

Document: 8, Title: Zoon Bin Laden wil in Groot Brittannie wonen

Document: 13, Title: Bin Laden junior wil dat pa ermee kapt

Cluster 1

Document: 1, Title: Anderlecht verhuurt Cyril Thereau aan Charleroi

Document: 12, Title: Anderlecht huurt Yakovenko

Document: 16, Title: Anderlecht strikt Guillaume Gillet

Document: 18, Title: Luigi Pieroni naar Anderlecht

Cluster 2

Document: 2, Title: McCanns onderhandelen over verfilming Maddie

Document: 3, Title: McCanns verdenken politie onderzoek te vertragen

Document: 14, Title: McCanns verspreiden foto van mogelijke ontvoerder

Document: 19, Title: Foto van kidnapper is afleidingstruc van McCanns

Cluster 3

Document: 4, Title: Angelina is zwanger geen twijfel mogelijk

Document: 6, Title: Angelina zwanger van tweeling

Document: 10, Title: Jurk Angelina wakkert geruchten over zwangerschap aan

Document: 11, Title: Angelina Jolie en Brad Pitt oefenen voor tweede kind

Cluster 4

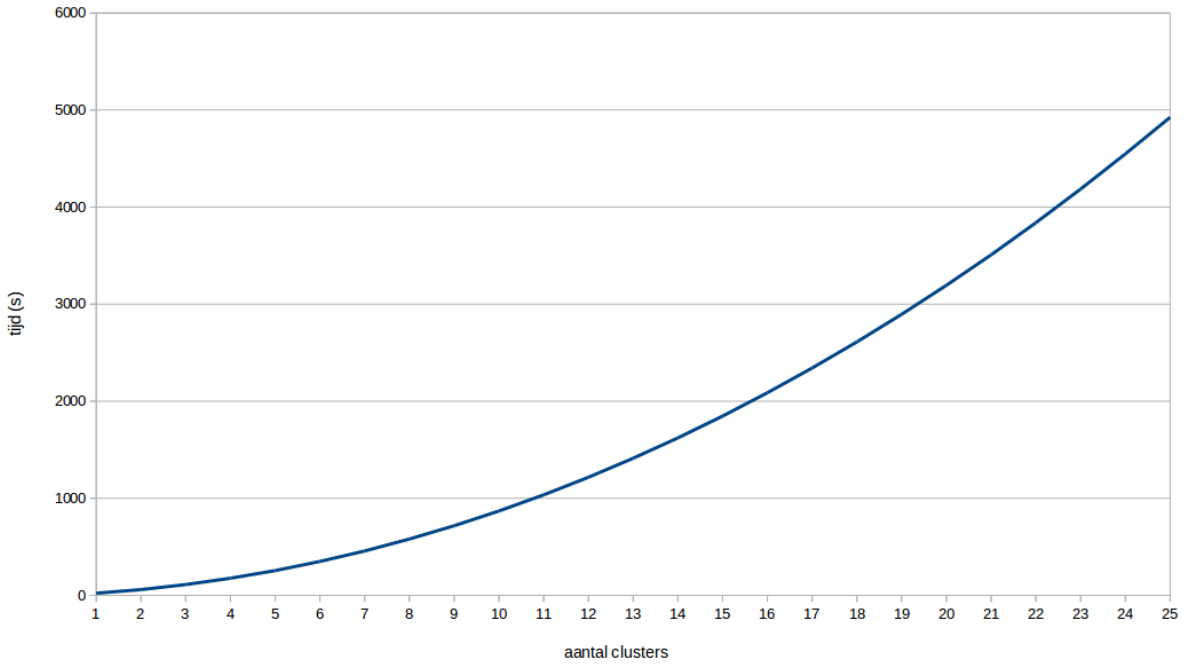
Document: 7, Title: Analyse waarom South Carolina Obama zuur kan opbreken
Document: 9, Title: Clinton en Obama strijden in South Carolina
Document: 15, Title: Clinton en McCain op kop in race naar Witte Huis
Document: 17, Title: Analyse Wie is John McCain

Door het gebruik van NEs wordt de vooropgestelde clustering perfect gevonden in de kleine dataset. Deze clustering wordt vervolgens getest op een grotere dataset in de volgende paragraaf.

Resultaten voor grote datasets

Het clusteralgoritme met NLP wordt toegepast op een dataset met 10121 documenten (Het Laatste Nieuws, januari 2008). De validity measure wordt op 0,32 gezet en het algoritme wordt uitgevoerd met iteratieve k (1000 features, 100 iteraties om cluster centra te bepalen). Van bij het begin kunnen we opmerken dat dit algoritme niet zal kunnen gebruikt worden voor grote hoeveelheden data. De totale tijd die nodig is om clusters te maken stijgt bijna exponentieel en neemt al snel een uur in beslag om 25 clusters te vinden (zie afbeelding 5.4). Op dat moment krijgt de clustering een validity measure van 0,93 en zijn de clusters dus nog niet gevormd.

Figuur 5.4: Aantal clusters in het iteratieve k-means algoritme in relatie tot het aantal clusters voor een dataset met 10121 documenten



Om de totale tijd die het algoritme nodig heeft te doen dalen worden twee implementaties

voorgesteld. De eerste in Apache Spark (zie 5.3.4) en een tweede in Apache Mahout (zie 5.3.5). Beiden werken bovenop het Hadoop filesysteem om sneller te kunnen werken en mogelijk gemakkelijker te schalen bij groter wordende datacollecties. Bovendien ondersteunen beide systemen enkele vormen van clustering.

5.3.4 Implementatie van K-Means met Apache Spark

Apache Hadoop [1] is een open-source framework dat oplossingen biedt voor big data, samen met uitgebreide verwerkings- en analysetools. Hadoop bestaat uit twee grote componenten: HDFS (Hadoop Distributed File System) en MapReduce [9]. HDFS is een schaalbaar, efficiënt en replica-based opslagmedium voor data. Het is gebaseerd op een master-slave architectuur waar "namenode" de master is en de "datanodes" de slaves. De slaves bevatten ook de effectieve data.

MapReduce zorgt ervoor dat gerepliceerde data snel en parallel verwerkt kan worden volgens map en reduce fases. *Map* is de fase die gebruikt wordt om in parallel de delen van het gedistribueerd systeem op te halen via verschillende *mappers*. De outputs van de mappers wordt door sorteer- en shufflealgoritmes gehaald om in de *reduce* fase de data te aggregeren en het resultaat te vinden van het initieel probleem.

Apache Spark werd ontwikkeld in het UC Berkeley AMPLab en werd open-source in 2010. Het is ontworpen voor efficiënte gegevensverwerking en bied een gebruiksvriendelijke interface aan om zo tot betere oplossingen te komen voor big data toepassingen. Spark heeft sinds 2012 bovendien een set robuuste en schaalbare leeralgoritmes onder de naam "MLlib" aan boord en ondersteunt een veel breder gamma aan applicaties dan MapReduce, maar behoud daarbij wel zijn automatische fouttolerantie.

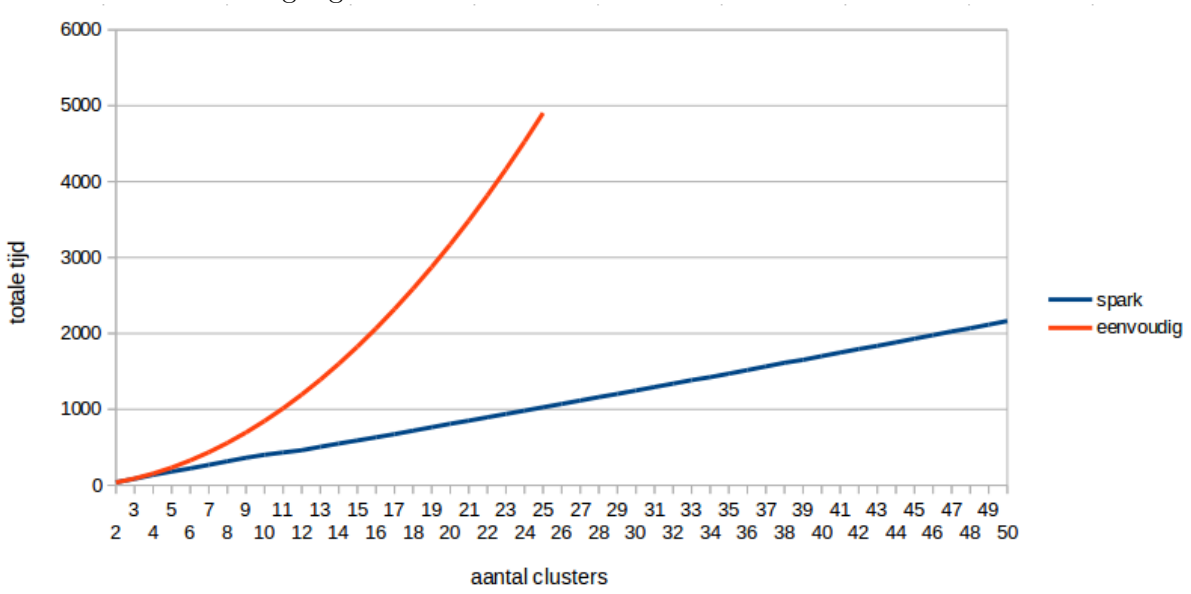
Spark werkt met RDDs (Resilient distributed Datsets) om efficiente applicaties te ondersteunen. RDDs kunnen opgeslagen worden in het geheugen tussen query's zonder dat replicatie nodig is. Indien data verloren gaat wordt deze opnieuw opgebouwd gebruik makend van *lineage*: elke RDD herinnert hoe het gebouwd is van een andere dataset en kan zichzelf zo herbouwen. Op die manier presteert Spark beter dan alle bestaande modellen.

Apache Spark biedt in zijn MlLib een k-means algoritme aan. Dit werd getest in [16] en werkt tot drie keer sneller dan hetzelfde algoritme gebruik makend van MapReduce. De implemenatie is hier wel zo dat er geen gebruik kan gemaakt worden van een alternatieve afstandsfunctie.

Om voor de snelheid van Apache Spark te kiezen zijn we dus verplicht over te stappen naar de Euclidische afstandsfunctie.

Een vergelijking in tijd om de eerste 25 clusters te maken bij de k-means implementatie van Apache Spark (met 1000 features en 100 iteraties) en het eenvoudige algoritme dat hiervoor reeds gebruikt werd kan gevonden worden in figuur 5.5.

Figuur 5.5: Vergelijking tussen de tijd om k-means clustering door te voeren in Apache Spark en binnen een eenvoudig algoritme



De grafiek van het eenvoudige algoritme is dezelfde als in figuur 5.4, maar we merken vooral de quasi lineaire tijd van Apache Spark op. Deze doet er gemiddeld 44 seconden over om een clustering uit te voeren in de eerste 50 clusters, terwijl het eenvoudige algoritme al 377 seconden nodig had om cluster 25 te vormen. Door Apache Spark te gebruiken kan de rekentijd dus flink beperkt worden, maar ten koste van het gebruik van de Cosine distance. Om de efficiëntie van dit algoritme te combineren met de positieve invloed van de Cosine distance wordt een nieuwe implementatie in Apache Mahout voorgesteld in de volgende sectie.

5.3.5 Implementatie van K-Means met Apache Mahout

Apache Mahout [35] is een open-source machine learning library van Apache. Het ondersteunt recommender engines met collaborative filtering, clustering en classificatie. Ook is het erg schaalbaar en werkt (zoals Apache Spark) bovenop Apache Hadoop. Bovendien wordt Mahout als Java library aangeboden, waardoor deze eenvoudig in de workflow kan worden opgenomen.

Mahout biedt een methode om K-Means clustering te verbeteren aan de hand van *Canopy Centers*. Hiervoor werd de optimale k steeds bepaald door alle waarden voor k iteratief af te lopen en op basis daarvan de optimale waarde te selecteren. Alhoewel deze methode met enige zekerheid een optimale k zal vinden, is dit zeker niet snel en bijgevolg moeilijk bruikbaar voor zeer grote datasets. Dat probleem wordt natuurlijk evenredig groter met een groter wordende dataset.

De implementatie van het Canopy algoritme kan gevonden worden in [30]. Het basis-idee is dat het aantal berekeningen die nodig zijn om afstanden te berekenen sterk kan verminderen door de data eerst te verdelen in overlappende subsets en daarop de afstanden te berekenen tussen paren van items die aan dezelfde subset toebehoren. De sterkte van het algoritme ligt vooral in de snelheid waarmee clusters gevormd worden. Die sterkte is ook meteen de grootste zwakte van het algoritme, want de clusters die gevormd worden bij het Canopy algoritme zijn niet altijd zeer accuraat. Toch kan op deze manier het optimale aantal clusters k bepaald worden. Die k clusters dienen vervolgens als input voor het K-Means algoritme dat wel in staat is om de goede clusters te vinden.

Het Canopy algoritme moet eerst notie krijgen van hoe groot de clusters zijn waar naar op zoek gegaan wordt. Daarvoor worden twee variabelen geïntroduceerd: T_1 en T_2. Dit zijn twee tresholds voor afstand met $T_1 > T_2$. Het algoritme begint met een dataset van punten en lege lijst *canopies*. Er wordt geïtereerd over de dataset en bij elke iteratie wordt een punt van de dataset verwijderd en wordt een canopy aan de lijst toegevoegd met dat punt als centrum. Vervolgens worden de rest van de punten een voor een afgelopen en wordt telkens de afstand van dat punt tot alle canopy centra berekend. Als die afstand binnen T_1 valt, wordt dit punt toegevoegd aan de canopy. Als de afstand bovendien binnen T_2 valt wordt het verwijderd van de lijst. Dit om te voorkomen dat een nieuwe canopy zou aangemaakt worden in de opeenvolgende loops. Dit proces loopt tot de lijst leeg is.

Om te bepalen wat een goede waarde is voor T_1 en T_2 wordt nog even teruggegrepen naar de dataset met 20 artikels. In die dataset moeten 5 clusters gevormd worden. Om dat te bereiken wordt $T1$ gevarieerd van 0 tot 1 met iteraties van 0,01 (de cosine distance geeft immers enkel waarden aan tussen 0 en 1). Telkens wordt daarbij ook de bijhorende T_2 geïtereerd ($T_2 \leq T_1$ en per 0,01). De output van dit algoritme wordt vervolgens als input voor K-Means gebruikt en levert een bepaalde validity measure op. De belangrijkste resultaten zijn te vinden in tabel 5.9. In die tabel worden alle waarden voor T_1 en T_2 opgesomd die de gezochte clustering opleveren

voor deze kleine dataset.

Niet alle opgesomde waarden voor T_1 en T_2 leveren de optimale (correcte) clustering op. De waarden met een (*) geven wel een aanvaardbare clustering, maar splitsen de topic rond het presidentschap in Amerika nog verder op in Obama en Clinton enerzijds en McCain anderzijds. Omdat deze clustering heel specifiek is, terwijl die twee clusters toch heel nauw verwant zijn krijgen we daar een minder goede validity measure. Ook is het wenselijk dat de cluster deze artikels samen zou nemen, aangezien een gebruiker die geïnteresseerd is in een artikel rond Obama hoogstwaarschijnlijk ook zal geïnteresseerd zijn in de andere presidentskandidaten.

Mahout laat ook toe om de belangrijkste termen per cluster op te sommen. Dit gegeven kan mogelijk ook interessant zijn voor de recommender. Door deze belangrijke termen te verzamelen kan wat een gebruiker belangrijk vindt ook toegevoegd worden aan de vergelijking die uitzoekt welke artikels een gebruiker mogelijk interessant kan vinden. Enkele belangrijke termen voor het topic rond de Amerikaanse presidentsverkiezingen zijn bijvoorbeeld de volgende: carolina, obama, zwart, mccain, senator, clinton, peiling. Dit is een goede combinatie van sterke keywords en named entities die ons in staat kunnen stellen om andere artikels gemakkelijk aan te bevelen.

Vervolgens wordt deze methode toegepast op de dataset van Het Laatste Nieuws uit Januari 2008 (10121 documenten). De waardes voor de canopy worden gekozen gelijkaardig aan die uit tabel 5.9 ($T_1 = 0,96$ en $T_2 = 0,92$). De uitvoeringstijd van dit algoritme blijft onder de 15 minuten (inclusief preprocessing voor 10121 documenten). Deze clustering levert 1685 clusters op, met dus een gemiddelde inhoud van ongeveer 6 artikels per cluster.

5.3.6 Fuzzy k-means clustering

Fuzzy k-means laat toe om aan *overlapping* clustering te doen in plaats van *exclusive* clustering. Terwijl k-means de harde clusters zoekt (waar elk punt aan één cluster toebehoort), kan fuzzy k-means soft clusters ontdekken. Items behoren daarbij aan meerdere clusters toe met een bepaalde zekerheid. Deze zekerheid is een uitdrukking van de afstand die een punt heeft tot het centrum van een cluster.

Fuzzy k-means is een algoritme dat beschikbaar is in Apache Mahout. Dit algoritme neemt een belangrijke parameter: de *fuzziness factor* m (groter dan 1). Stel dat voor een vector V de afstanden tot elke van de k clusters gegeven zijn door $d_1, d_2, ..., d_k$. De fuzziness factor wordt dan gebruikt bij het bepalen van de graad van verbinding u_1 van een vector V tot de eerste

Tabel 5.9: Iteratie van de treshold parameters voor Canopy clustering (T_1 en T_2) die voor een documentenset van 20 documenten de 5 topics al dan niet correct detecteren met het k-means algoritme

T_1	T_2	clusters	validity	correct
0.96	0.92	5	0.64	ja
0.96	0.93	5	0.64	ja
0.96	0.94	5	0.67	nee
0.96	0.95	5	0.67	nee
0.97	0.92	5	0.64	ja
0.97	0.93	5	0.64	ja
0.97	0.94	5	0.68	ja
0.97	0.95	5	0.68	ja
0.97	0.96	5	0.68	ja
0.98	0.92	5	0.68	ja
0.98	0.93	5	0.68	ja
0.98	0.94	5	0.71	nee
0.98	0.95	5	0.71	nee
0.98	0.96	5	0.71	nee
0.98	0.97	5	0.71	nee
0.99	0.92	5	0.70	nee (*)
0.99	0.93	5	0.70	nee (*)
0.99	0.94	5	0.70	nee
0.99	0.95	5	0.70	nee
0.99	0.96	5	0.70	nee
0.99	0.97	5	0.70	nee
1	0.92	5	0.68	nee
1	0.93	5	0.68	nee

cluster C_1:

$$u_i = \frac{1}{\frac{d_1}{d_1}^{\frac{2}{m-1}} + \frac{d_1}{d_2}^{\frac{2}{m-1}} + ... + \frac{d_1}{d_k}^{\frac{2}{m-1}}} \tag{5.8}$$

De graad van verbinding tot een andere cluster kan gevonden worden door d_1 in de tellers van de uitdrukking in de noemer te vervangen door een andere cluster.

Enerzijds zullen bij een waarde 2 voor m alle graden van verbinding 1 worden. Anderzijds, als m dichter bij 1 komt zal meer belangrijkheid gegeven worden aan de cluster die het dichtst bij de vector ligt. Fuzzy k-means gedraagt zich bijgevolg meer als k-means wanneer de waarde voor m dicht bij 1 komt te liggen. Als m groter wordt zal de fuzziness van het algoritme verhogen en zal meer overlap merkbaar zijn.

Voor het bepalen van het aantal clusters wordt dezelfde methode gevolgd als bij k-means clustering (k zoeken via Canopy algoritme, clustering uitvoeren met Fuzzy k-means met dezelfde clustercentra). Fuzzy k-means steunt immers op hetzelfde principe als normale k-means.

Het voordeel om Fuzzy K-Means te gebruiken tegenover de normale k-means is dat een bepaald item beter gesitueerd kan worden. Een item dat gaat over ISIS, terrorisme en aanslagen kan zowel bij een cluster geplaatst worden rond de aanslagen in Brussel als bij een cluster rond de aanslagen in Parijs. Bij normale k-means zouden deze artikels misschien samen geclusterd worden, omdat de inhoud min of meer overeen komt. Met Fuzzy K-means kan de specificiteit van een cluster verhoogd worden om zo toch een onderscheid te maken tussen beide gebeurtenissen. Op die manier kan met kleinere topics dus soms meer informatie gehaald worden uit dezelfde data. Dit is dus zeker interessant in het kader van aanbevelingen. De gebruiker kan zo tussen verschillende aaneensluitende topics “springen” die hem waarschijnlijk ook zullen interesseren.

Als we Fuzzy K-Means toepassen op de kleine dataset van 20 documenten krijgen we eveneens 6 clusters (gebruik makend van dezelfde waarden voor canopy clustering uit de vorige sectie). De resultaten zijn echter minder kwalitatief dan bij normale k-means. Een artikel over Osama bin Laden wordt zo toch met een hoge waarschijnlijkheid geclusterd bij een artikel van de zwangerschap van Angelina Jolie. Als we dit bekijken in termen van een aanbevelingssysteem, dan kunnen we deze clustering niet toestaan. Het is de bedoeling dat gebruikers artikels krijgen aangeraden die zeer relevant zijn tegenover hun huidige voorkeuren. Bij fuzzy k-means is dit niet gegarandeerd.

5.3.7 Conclusie voor clustering

Door de schaalbaarheid naar meerdere machines, uitbreidbaarheid van de algoritmes en algemene snelheid van Apache Mahout framework wordt sowieso voor deze implementatie gekozen. Het clusteralgoritme dat gebruikt wordt in dit systeem wendt het canopy algoritme aan en voedt daarmee het normale k-means algoritme met het optimale aantal clusters. De implementatie van het k-means algoritme maakt gebruik van de cosine distance om gerelateerde arikels te zoeken. Dit wordt gecombineerd met de opwaarding van named entities om in een goede clustering te voorzien.

5.4 Aanbevelingssysteem

Het gebruikte aanbevelingssysteem is een klassieke content-based recommender, gebouwd in Lenskit [10]. In plaats van de manueel ingegeven tags wordt gebruik gemaakt van de verworven informatie uit zowel classificatie als clustering. Deze informatie wordt doorgegeven aan de recommender onder vorm van tags. Zo krijgt elk item een tag voor de hoofdcategorie, een tag voor de subcategorie en een of meerdere tags met de cluster of clusters waartoe het item behoort. Met deze data kan het content-based systeem gelijkaardige items vinden voor een bepaalde gebruiker. De resultaten en performantie van dit aanbevelingssysteem worden in hoofdstuk **??** besproken.

Hoofdstuk 6

Resultaten van het nieuwe aanbevelingssysteem

Zoals reeds besproken in 2.4 is een content-based recommender afhankelijk van tags die aan een bepaald artikel hangen. Deze data wordt in dit systeem aangemaakt door een combinatie van classificatie en clustering toe te passen. Om te testen of dit algoritme beter presteert dan een baseline recommender worden enkele testen uitgevoerd in dit hoofdstuk.

6.1 Baseline recommender systems

Om de resultaten van het nieuw ontworpen systeem te vergelijken met bestaande content-based recommender systemen, wordt gebruik gemaakt van een baseline recommender. Deze is gebaseerd op term frequency-inverse document frequency (zie vergelijking 5.1). In deze recommender worden de teksten verwerkt met tf-idf toe te passen op de volledige inhoud om zo de meest relevante artikels aan te raden aan gebruikers.

6.2 Vergelijking van het baseline recommenders systeem en de nieuw ontworpen recommender

Om een objectieve vergelijking uit te voeren tussen de drie systemen worden enkele waarden gebruikt. De eerste daarvan is root-mean-square error (RMSE [23]). Dit is misschien de meest populaire meetwaarde voor evaluatie van voorspelde ratings [44]. Ht systeem genereert ratings

$\hat{r}_{ui}$ voor een testset T van user-item paren (u, i) waarvoor de echte ratings gekend zijn. De RMSE wordt dan gegeven door:

$$RMSE = \sqrt{\frac{1}{|T|} \sum_{(u,i) \in T} (\hat{r}_{ui} - r_{ui})^2} \tag{6.1}$$

Een tweede waarde bepaalt de *coverage* [19] van een recommender. Dit is een waarde die bepaalt hoeveel gebruikers op zijn minst één aanbeveling krijgen en hoeveel van de items in de catalogus kan worden aanbevolen. Systemen met lagere coverage zijn minder aantrekkelijk voor gebuikers omdat ze gelimiteerd zijn in problemen waarmee ze behulpzaam kunnen zijn.

Bij de derde waarde, nDCG (normalized Discounted Cumulative Gain) [6], worden twee veronderstellingen gemaakt:

1. Zeer relevante documenten zijn beter bruikbaar wanneer ze eerder in de resultaten terugkomen (ze krijgen een hogere ranking)

2. Zeer relevante documenten zijn beter bruikbaar dan minder relevante documenten, welke op hun beurt beter bruikbaar zijn dan niet-relevante documenten.

nDCG is een meetwaarde die posities logaritmisch verdisconteerd. Als uitgegaan wordt dat een gebruiker u een "gain" g_{ui} kan hebben door item i aanbevolen te krijgen, dan wordt de gemiddelde DCG (Discounted Cumulative Gain) voor een lijst van J items gegeven door:

$$DCG = \frac{1}{N} \sum_{u=1}^{N} \sum_{j=1}^{J} \frac{g_{ui_j}}{max(1, \log_b j)} \tag{6.2}$$

De parameter log_b is daarbij een vrije parameter die van $\log_2$ tot $\log_{10}$ kan gaan. Hier wordt $\log_2$ gebruikt. nDCG is de genormaliseerde versie van DCG:

$$NDCG = \frac{DCG}{DCG*} \tag{6.3}$$

waarbij DCG* gelijk staat aan de ideale DCG.

6.3 Resultaten

Om deze resultaten te evalueren werd de evaluatietoolkit van Lenskit gebruikt. De dataset die gebruikt wordt is die van Het Laatste Nieuws uit januari 2008. Het grootste probleem bij deze content-based recommenders is dat ze steunen op de artikels die een gebruiker reeds een rating gegeven heeft.

Alle meetwaarden werden geverifieerd aan de hand van een 10-fold cross-validation. De resultaten zijn terug te vinden in tabel **??**.

Hoofdstuk 7

Conclusie

7.1 Algemeen

De doelstelling van deze masterproef is om aanbevelingen te genereren voor gebruikers gebaseerd op de werkelijke inhoud van die items. In dit onderzoek worden daarom twee vormen van content recognition gebruikt, namelijk classificatie en clustering. Een korte literatuurstudie over deze methodes en over de gebruikte aanbevelingstechnieken is te vinden in hoofdstuk 2.

Hoofdstuk 3 bouwt verder op deze systemen van content recognition en voorziet in een architectuur om beide vormen te combineren. Het belangrijkste gegeven hier is dat classificatie en clustering apart worden uitgevoerd, elk met hun eigen fase van preprocessing. Op die manier kunnen verschillende algoritmes ingezet worden met elk hun eigen voordelen om tot betere classificatie en clustering te komen.

In hoofdstuk 4 wordt de effectieve implementatie beschreven, samen met enkele andere overwogen opties en resultaten van verschillende benchmarks op zoek naar de beste prestaties. Voor classificatie werd gevonden dat de optimale classifiers per dataset kunnen verschillen. Om binnen een dataset de structuur van classificatie te kunnen volgen wordt per opdeling in categorieën een aparte classifier gekozen. Ook deze hangt steeds af van de moeilijkheid van het classificatieprobleem. Daarbij is het aantal woorden die in het woordenboek van de classifier opgenomen wordt van groot belang, evenals het algoritme dat door deze classifiers word gevolgd. Om de optimale classifier te vinden voor elke categorie in de dataset werd gebruik gemaakt van het WEKA framework.
Voor clustering werd een systeem opgesteld in Apache Mahout gebruik makend van k-means

clustering. Mahout zorgt ervoor dat zelfs voor een grote dataset de clustering op een vlotte manier kan verlopen. Ook werd om de clustering te verbeteren gebruik gemaakt van named entity recognition. Door de namen van personen, plaatsen en organisaties een hogere belangrijkheid te geven binnen een tekst kan de clustering verbeterd worden. Ook helpt het de gebruiker omdat clusters die rond named enitities gevormd worden beter te begrijpen zijn voor gebruikers van aanbevelingssystemen.

In hoofdstuk 5 worden enkele testen overlopen die de kwaliteit van het aanbevelingssysteem onder de loep nemen.

Bibliografie

[1] Apache Hadoop Documentation.

[2] Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6):734–749, 2005.

[3] Yoshua Bengio and Yves Grandvalet. No Unbiased Estimator of the Variance of K-Fold Cross-Validation. *Journal of Machine Learning Research*, 5:1089–1105, 2004.

[4] James Bennett and Stan Lanning. The Netflix Prize. *KDD Cup and Workshop*, pages 3–6, 2007.

[5] Toine Bogers and Antal Van Den Bosch. *Collaborative and content-based filtering for item recommendation on social bookmarking websites*, volume 532. 2009.

[6] Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. Learning to rank using gradient descent, 2005.

[7] Xiao Cheng and Dan Roth. Relational Inference for Wikification. *Empirical Methods in Natural Language Processing*, (October):1787–1796, 2013.

[8] Nancy Chinchor and Patty Robinson. MUC-7 Named Entity Task Definition. *Proceedings of the Sixth Message Understanding Conference MUC6*, (September):21, 1997.

[9] Jeffrey Dean and Sanjay Ghemawat. MapReduce: Simplied Data Processing on Large Clusters. *Proceedings of 6th Symposium on Operating Systems Design and Implementation*, pages 137–149, 2004.

[10] Michael D Ekstrand, Michael Ludwig, Joseph a Konstan, and John T Riedl. Rethinking the Recommender Research Ecosystem : Categories and Subject Descriptors. *Proceedings of the 5th ACM conference on Recommender systems - RecSys '11*, pages 133–140, 2011.

[11] UM Fayyad, G Piatetsky-Shapiro, and P Smyth. Knowledge Discovery and Data Mining: Towards a Unifying Framework. *Proc 2nd Int Conf on Knowledge Discovery and Data Mining Portland OR*, pages 82–88, 1996.

[12] Jenny Rose Finkel, Trond Grenager, and Christopher Manning. Incorporating non-local information into information extraction systems by gibbs sampling. *in Acl*, (1995):363 – 370, 2005.

[13] George Forman. An Extensive Empirical Study of Feature Selection Metrics for Text Classification. *Journal of Machine Learning Research*, 3:1289–1305, 2003.

[14] N Friburger and D Maurel. Textual Similarity Based on Proper Names. *Proceedings of the workshop {Mathematical/Formal} Methods in Information Retrieval {(MFIR'2002)} at the 25 {thACM} {SIGIR} Conference*, pages 155–167, 2002.

[15] Alfio Gliozzo and Carlo Strapparava. Cross language text categorization by acquiring multilingual domain models from comparable corpora. *Proceedings of the ACL Workshop on Building and Using Parallel Texts*, (June):9–16, 2005.

[16] Satish Gopalani and Rohan Arora. Comparing Apache Spark and Map Reduce with Performance Analysis using K-Means. 113(1):8–11, 2015.

[17] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. The WEKA data mining software. *ACM SIGKDD Explorations*, 11(1):10–18, 2009.

[18] Jiawei Han, Micheline Kamber, and Jian Pei. *Data Mining: Concepts and Techniques*. 3 edition edition, 2012.

[19] Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen, and John T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS)*, 22(1):5–53, 2004.

[20] Andreas Hotho, Andreas Nürnberger, and Gerhard Paaß. A Brief Survey of Text Mining. *LDV-Forum*, 20(1):19–62, 2005.

[21] Huan Liu and Lei Yu. Toward integrating feature selection algorithms for classification and clustering. *IEEE Transactions on Knowledge and Data Engineering*, 17(4):491–502, apr 2005.

[22] Anna Huang. Similarity measures for text document clustering. *Proceedings of the Sixth New Zealand*, (April):49–56, 2008.

[23] Rob J Hyndman and Anne B Koehler. Another look at measures of forecast accuracy. 22:679–688, 2006.

[24] Josef Kittler, Ieee Computer Society, Mohamad Hatef, Robert P W Duin, and Jiri Matas. On Combining Classifiers. 20(3):226–239, 1998.

[25] Trupti Kodinariya and Prashant Makwana. Review on determining number of Cluster in K-Means Clustering. *International Journal of Advance Research in Computer Science and Management Studies*, 1(6):90–95, 2013.

[26] Wessel Kraaij and Renée Pohlmann. Porter's stemming algorithm for Dutch. *Informatiewetenschap 1994: Wetenschapelijke bijdragen aan de derde STINFON Conferentie*, pages 167–180, 1994.

[27] Aristidis Likas, Nikos Vlassis, and Jakob Verbeek. The global k-means clustering algorithm Intelligent Autonomous Systems. *ISA technical report series*, 2011.

[28] G. Linden, B. Smith, and J. York. Amazon.com recommendations: item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1):76–80, jan 2003.

[29] Stuart P. Lloyd. Least Squares Quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.

[30] Andrew McCallum, Kamal Nigam, and Lyle H Ungar. Efficient clustering of high-dimensional data sets with application to reference matching. *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining KDD 00*, pages 169–178, 2000.

[31] Paul McNamee and James Mayfield. Character N-Gram Tokenization for European Language Text Retrieval. *Information Retrieval*, 7(1-2):73–97, 2004.

[32] David Milne, Olena Medelyan, and Ian H. Witten. Mining domain-specific thesauri from Wikipedia: A case study. *Proceedings - 2006 IEEE/WIC/ACM International Conference on Web Intelligence (WI 2006 Main Conference Proceedings), WI'06*, pages 442–448, 2007.

[33] Soto Montalvo, Raquel Martínez, Víctor Fresno, and Agustín Delgado. Exploiting named entities for bilingual news clustering. *Journal of the Association for Information Science and Technology*, 66(2):363–376, feb 2015.

[34] Divya Nasa. Text Mining Techniques- A Survey. 2(4):1–5, 2012.

[35] Sean Owen, Robin Anil, Ted Dunning, and Ellen Friedman. *Mahout in Action*. 2011.

[36] M.F. Porter. An algorithm for suffix stripping, 1980.

[37] David M W Powers. Evaluation: From Precision, Recall and F-Factor to ROC, Informedness, Markedness & Correlation. (December):24, 2007.

[38] Siddheswar Ray and Rose H Turi. Determination of Number of Clusters in.

[39] Van V. J. Rigsbergen. A non classical logic for Information retrieval. *The Computer Journal*, 29(6):481–485, jun 1986.

[40] S.E. ROBERTSON. the Probability Ranking Principle in Ir, 1977.

[41] G Salton, A Wong, and C S Yang. A vector space model for automatic indexing. *Cacm*, 18(11):613–620, 1975.

[42] Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, 24(5):513–523, jan 1988.

[43] AK Seewald and J Fürnkranz. An evaluation of grading classifiers. *Advances in Intelligent Data Analysis. Lecture Notes in Computer Science*, 2189:115–124, 2001.

[44] Guy Shani and Asela Gunawardana. Evaluating recommendation systems. *Recommender systems handbook*, pages 257–298, 2011.

[45] Georgios Sigletos, Georgios Paliouras, Constantine D Spyropoulos, and Michalis Hatzopoulos. Combining Information Extraction Systems Using Voting and Stacked Generalization Georgios Sigletos Georgios Paliouras. *Journal of Machine Learning Research*, 6:1751–1782, 2005.

[46] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. Chap 8 : Cluster Analysis: Basic Concepts and Algorithms. *Introduction to Data Mining*, page Chapter 8, 2005.

[47] The Apache Software Foundation. Apache Tika.

[48] R Tibshirani, G Walther, and T Hastie. Estimating the number of clusters in a data set via the gap statistic, 2001.

[49] Timothy Weale, Chris Brew, and Eric Fosler-lussier. Using the wiktionary graph structure for synonym detection. *In Proceedings of the Workshop on the People's Web Meets NLP*, (August):28–31, 2009.

[50] Yorick Wilks. Information extraction as a core language technology. pages 1–9. 1997.

[51] Torsten Zesch, Christof Müller, and Iryna Gurevych. Using Wiktionary for Computing Semantic Relatedness. *Proceedings of AAAI*, 8:861–866, 2008.

Lijst van figuren

Lijst van tabellen