KATHOLIEKE UNIVERSITEIT
LEUVEN

Faculteit
Ingenieurswetenschappen

Uitbreiding van N-gram taalmodellen met semantische informatie

Bruno De Laet

Thesis voorgedragen tot het
behalen van de graad van Master
in de ingenieurswetenschappen:
wiskundige ingenieurstechnieken

Promotor:
Prof. dr. ir. Patrick Wambacq

Assessoren:
Prof. dr. ir. Johan Suykens
Prof. dr. ir. Marc Van Barel

Begeleiders:
Dr. ir. Kris Demuynck
Joris Pelemans

Academiejaar 2011 – 2012

Voorwoord

Deze masterproef vormde een belangrijk onderdeel binnen mijn laatste opleidingsjaar. Het werk zou niet tot stand zijn gekomen zonder al de mensen die mij hierbij geholpen hebben.

In de eerste plaats zou ik mijn promotor professor Patrick Wambacq willen bedanken. Vanaf onze eerste bijeenkomst kreeg ik meteen een positief beeld over deze masterproef. Tijdens alle volgende bijeenkomsten kreeg ik van de professor steeds de nodige hulp en feedback om dit project tot een goed einde te brengen.

Voorts zou ik ook mijn begeleiders Kris Demuynck en Joris Pelemans willen bedanken. Zij waren steeds bereid om al mijn vragen te beantwoorden. Joris wil ik vooral ook bedanken voor het uitvoerig verbeteren van de tekst en het artikel, en Kris voor het helpen bij technische moeilijkheden tijdens het implementeren.

Daarnaast wil ik ook de assessoren professor Marc Van Barel en professor Johan Suykens bedanken voor het aanwezig zijn op de presentaties en het lezen van de tekst.

Tenslotte zou ik ook nog mijn naaste omgeving willen bedanken. In de eerste plaats mijn familie: mijn mama en papa, mijn zus en vriend en nichtje, en mijn broer en schoonzus. Verder zou ik ook mijn vrienden niet willen vergeten, met in het bijzonder Roeland, die vele werkuren naast mij in de computerklassen van het departement heeft doorgebracht.

Bruno De Laet

Inhoudsopgave

Samenvatting

De verwerking van natuurlijke taal wordt steeds belangrijker. Een voorbeeld is het dicteren van een tekst aan een computer. Een automatische spraakherkenner zal trachten om gesproken taal in tekst om te zetten. Dit proces verloopt in verschillende stappen. Een onderdeel van een automatische spraakherkenner is het taalmodel. Het taalmodel geeft de a priori kans van een woordsequentie terug. Het doel van deze masterproef is om een geschikt taalmodel te vinden, en dit voor het Nederlands. Dit taalmodel kan dan geïntegreerd worden in een automatische spraakherkenner.

Gestart is met de meest gebruikte taalmodellen, N-grammen. Deze modellen voorspellen het volgende woord op basis van de $N-1$ vorige woorden. Het aanmaken van het model gebeurt op basis van frequentietellingen op grote hoeveelheid tekstdata. Smoothing en backoff/interpolatie zijn technieken om met de spaarsheid van N-grammen om te gaan.
Cachingmodellen zijn gebaseerd op het feit dat pas voorgekomen woorden een grote kans hebben om weldra nog eens voor te komen. Omdat deze modellen nog spaarser zijn dan N-grammen, worden ze dikwijls met N-grammen gecombineerd.

In een volgend hoofdstuk is latent semantische analyse (LSA) besproken. LSA gaat ervan uit dat gelijkaardige woorden in gelijkaardige documenten voorkomen en gaat op zoek naar verborgen betekenisovereenkomsten tussen woorden. Het LSA-taalmodel wijst een kans toe aan het volgende woord op basis van de afstand tussen dit woord en de documentgeschiedenis. Omdat LSA-taalmodellen lokale afhankelijkheden binnen taal negeren, worden ze dikwijls gecombineerd met N-grammen.

Daarna komt probabilistische latent semantische analyse (pLSA) aan bod. Deze techniek werkt enkele tekortkomingen van LSA weg. Het werkt in tegenstelling tot LSA vanuit een statistisch oogpunt. Het topic model gaat ervan uit dat documenten een mengeling zijn van topics, die zelf een mengeling van woorden zijn. Op basis van de aanwezige topics in de documentgeschiedenis zal het pLSA-taalmodel de kans op het volgende woord teruggeven. Ook hier wordt het taalmodel meestal gecombineerd met N-grammen.

In de experimenten zijn de verschillende taalmodellen onderling vergeleken. In een eerste stadium worden alle modelparameters geoptimaliseerd door het minimaliseren van de perplexiteit op een validatieset. Nadien kan van elk finaal model

de perplexiteit berekend worden op twee onafhankelijke testsets. Elk taalmodel wordt vergeleken ten opzichte van het basis N-gram taalmodel. Het LSA-taalmodel haalt duidelijk de beste resultaten. De resultaten van het cachingmodel en het pLSA-taalmodel zijn vergelijkbaar.

Lijst van figuren

Lijst van tabellen

Hoofdstuk 1

Inleiding

Er is een steeds toenemende behoefte aan de verwerking van natuurlijke taal. Een voorbeeld hiervan is het dicteren van een tekst aan of het analyseren van gesproken tekst door een computer. Automatische spraakherkenning (Automatic Speech Recognition of ASR) is het vertalen door computers van gesproken taal in tekst. Als input in het systeem komt het spraaksignaal binnen, en het doel van automatische spraakherkenning is om de bijbehorende woordsequentie terug te geven. Daarbij streeft het naar een zo goed mogelijke nabootsing van de menselijke capaciteiten om spraak te herkennen. Dit is echter een zeer complex proces dat moeilijk met computers te modelleren is. De huidige spraakherkenners halen nog lang niet het herkenningsniveau van een doorsnee mens. Er is echter nog veel progressiemarge in deze technologie. Momenteel is er volop onderzoek naar betere spraakherkenners aan de gang.

1.1 Automatische spraakherkenning

Een overzicht van de werking van automatische spraakherkenning wordt gegeven in [29]. Ter ondersteuning wordt in figuur 1.1 het proces van automatische spraakherkenning schematisch afgebeeld. De input van de spraakherkenner is het spraaksignaal, afgebeeld bovenaan in het midden van de figuur. Uit het spraaksignaal wordt dan per tijdsframe de nuttige informatie opgeslagen in een akoestische vector y_i. Deze stap heet de front-end parametrisatie. De taak van de automatische spraakherkenner is nu om de woordsequentie $w_1 w_2 ... w_n = w_1^n$ te bepalen die het meest waarschijnlijk is, gegeven de akoestische vectoren $y_1 y_2 ... y_n = y_1^n$:

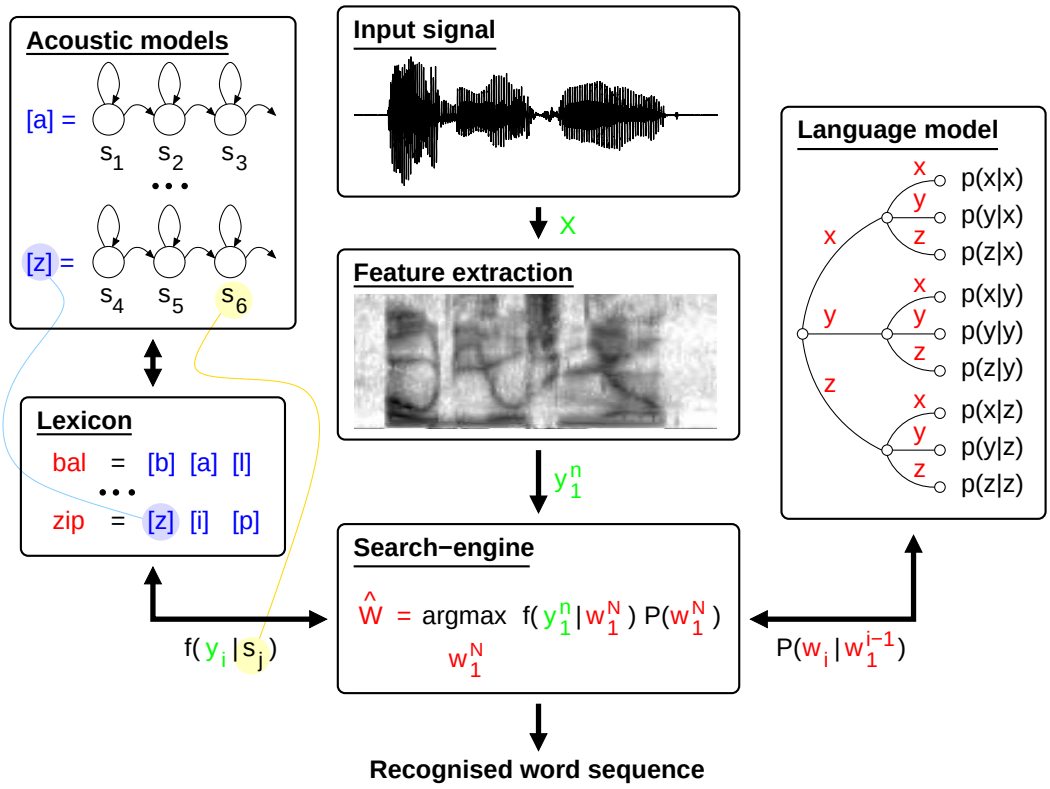
$$\hat{W} = \arg\max_{w_1^n} P(w_1^n | y_1^n) \tag{1.1}$$

Deze formule kan met de regel van Bayes herschreven worden als:

$$\hat{W} = \arg\max_{w_1^n} \frac{P(y_1^n | w_1^n) P(w_1^n)}{P(y_1^n)} \tag{1.2}$$

Aangezien de term $P(y_1^n)$ onafhankelijk is van w_1^n, wordt de meest waarschijnlijke woordsequentie gevonden uit de maximalisatie van de vermenigvuldiging van de

termen $P(y_1^n|w_1^n)$ en $P(w_1^n)$. De eerste term stelt de kans voor dat een bepaalde sequentie van akoestische vectoren is opgetreden, gegeven een woordsequentie. Dit is het akoestisch model en staat afgebeeld in het linkerdeel van figuur 1.1. De tweede term stelt de a priori kans voor dat een bepaalde woordsequentie optreedt. Deze kans wordt bepaald door het taalmodel, weergegeven in het rechterdeel van figuur 1.1.



Figuur 1.1: Overzicht van automatische spraakherkenning.

1.1.1 Front-end parametrisatie

De eerste stap in automatische spraakherkenning is het omzetten van het spraaksignaal in akoestische vectoren. Het spraaksignaal wordt eerst onderverdeeld in kleine blokken, die elk typisch ongeveer 25 msec lang zijn. Gedurende deze periode wordt het spraaksignaal constant verondersteld. Er is steeds een kleine overlap tussen opeenvolgende blokken en het is ook gebruikelijk om een vensterfunctie op elk blok uit te voeren.
Er bestaan verschillende methodes om de akoestische vectoren te berekenen. Een

veel gebruikte methode berekent de mel-frequentie cepstrale coëfficiënten. Eerst wordt van elk tijdsblok de fast-fouriertransformatie (FFT) berekend. In de volgende stap worden de bekomen coëfficiënten door zogenaamde melschaal driehoekige filters gehaald. Vervolgens wordt van elke coëfficiënt het logaritme genomen en de laatste stap is het uitvoeren van een discrete cosinus transformatie (DCT) op de filterbank coëfficiënten. De meeste spectrale informatie zit vervat in de lagere orde coëfficiënten en daarom worden enkel de 13 eerste coëfficiënten bijgehouden.
Aangezien opeenvolgende tijdsframes gecorreleerd zijn, worden aan de 13 coëfficiënten ook de eerste en tweede differenties tussen omliggende tijdsframes toegevoegd, wat een totaal maakt van 39 coëfficiënten per akoestische vector.

1.1.2 Akoestisch model

De taak van het akoestisch model is het berekenen van de kansen $P(y_1^n|w_1^n)$. Hierbij wordt eerst een woordsequentie w_1^n gepostuleerd. Elk woord wordt dan opgesplitst in basisgeluiden, zogenaamde *fonen*. De verzameling van woorden opgesplitst in fonen is het lexicon, zie figuur 1.1. Elke foon is voorgesteld door een verborgen Markovmodel (HMM), zie ook links bovenaan figuur 1.1. Een verborgen Markovmodel heeft een aantal toestanden die onderling verbonden zijn. In tegenstelling tot een gewoon Markovmodel zijn de toestanden niet direct observeerbaar. De emissie- en transitieprobabiliteiten worden geschat op basis van herkende data. De exacte werking van verborgen Markovmodellen in het akoestisch model valt buiten het bereik van deze masterproef en meer informatie is te vinden in [29].
De verborgen Markovmodellen van de opeenvolgende fonen worden aan elkaar geschakeld en vormen een samengesteld verborgen Markovmodel. Voor elke gepostuleerde woordsequentie is het nu mogelijk de kans $P(y_1^n|w_1^n)$ te bepalen door de sommatie te nemen over alle mogelijke toestandsovergangen in het verborgen Markovmodel. Een alternatief is om $P(y_1^n|w_1^n)$ gelijk te stellen aan de kans van de meeste waarschijnlijke toestandsovergang.

1.1.3 Taalmodel

De opdracht van het taalmodel bestaat erin de kans terug te geven dat een bepaald woord w_k optreedt, gegeven de vorige woorden w_1^{k-1}. In figuur 1.1 is de werking van het taalmodel geïllustreerd voor een vocabularium van 3 woorden x, y en z. Voor het eerste woord worden de kansen $P(x)$, $P(y)$ en $P(z)$ berekend. Als het eerste woord x was, worden voor het tweede woord de kansen $P(x|x)$, $P(y|x)$ en $P(z|x)$ berekend. Op deze manier is het mogelijk om elk van de kansen $P(w_k|w_1^{k-1})$ te bepalen.
Er bestaan zowel grammaticagebaseerde als zuiver statistische taalmodellen. Terwijl de eerste soort modellen nog trachten de grammaticaregels van taal te modelleren, houden de tweede soort modellen hier geen rekening meer mee. Taal ligt echter niet strikt vast en er zijn enorm veel mogelijkheden om binnen spraak een zin te maken. Dat maakt het moeilijk om de taal te definiëren volgens een vaste grammatica. Een ander nadeel van grammaticagebaseerde modellen is dat de modellen voor verschillende talen telkens opnieuw zorgvuldig moeten opgesteld worden. Een zuiver

statistisch taalmodel is daarentegen universeel, voor elke taal kan een zelfde soort modelvorm gebruikt worden. Het enige dat onderling tussen de modellen voor verschillende talen zal verschillen is de trainingsset om het model te trainen.

1.1.4 Decodering

Er is reeds uitgelegd hoe de afzonderlijke termen uit vergelijking 1.2 berekend kunnen worden. Het is echter nog niet duidelijk hoe nu juist de woordsequentie w_1^n wordt gevonden die de gecombineerde uitdrukking maximaliseert. Dit is de taak van de decoder. Een mogelijkheid is om alle mogelijke woordsequenties te postuleren en dan de woordsequentie met de hoogste waarde te kiezen. Dit is echter praktisch niet haalbaar. Heel veel van die woordsequenties zullen heel onwaarschijnlijk zijn. Het idee is om de oplossingenverzameling klein te houden. Voor het eerste te herkennen woord worden alle woorden uit het vocabularium gepostuleerd. De gecombineerde kans uit vergelijking 1.2 wordt dan voor elk woord berekend. De woorden waarvan de gecombineerde kans onder een vooraf te bepalen drempelwaarde vallen, worden uit de oplossingenverzameling verwijderd. Voor het tweede te herkennen woord worden vervolgens opnieuw alle woorden uit het vocabularium gepostuleerd en deze worden verbonden met alle overgebleven mogelijkheden voor het eerste woord. Dan wordt de gecombineerde kans berekend en de combinaties die onder de drempelwaarde vallen, worden opnieuw verwijderd. Op die manier blijft de oplossingenverzameling tijdens het decoderen klein genoeg. Dit proces gaat door tot het laatste te herkennen woord. Op het einde zal de decodering de woordsequentie met de hoogste gecombineerde kans teruggeven als herkende sequentie.

1.2 Doel masterproef

In deze masterproef ligt de focus op het vinden van een geschikt statistisch taalmodel, dat gebruikt kan worden binnen automatische spraakherkenning. Van de andere onderdelen binnen automatische spraakherkenning wordt abstractie gemaakt. Het resultaat van deze masterproef zal bijgevolg geen op zichzelf staande spraakherkenner zijn, maar zal een taalmodel zijn dat in combinatie met een front-end parametrisatie-model, akoestisch model en decoder een automatische spraakherkenner kan vormen. In de literatuur zijn al heel wat verschillende statistische taalmodellen beschreven. De meeste van deze modellen zijn getest op Engelstalige tekstdata. Deze masterproef zal onderzoeken hoe enkele van deze taalmodellen scoren op Nederlandstalige tekst-data. In hoofdstuk 2 zal gestart worden met de eenvoudige taalmodellen. Deze zijn N-grammen, de meest gebruikte taalmodellen, en cachingmodellen. Op het einde van dat hoofdstuk zal een afleiding gegeven worden om tot de evaluatie van een specifiek taalmodel te komen. In de hoofdstukken 3 en 4 komen twee technieken aan bod die leiden tot geavanceerde taalmodellen: latent semantische analyse en probabilistische latent semantische analyse. De combinatie van deze technieken met de eenvoudige N-grammen komt ook uitgebreid aan bod. Vervolgens worden in hoofdstuk 5 de experimenten beschreven, hierin worden alle modellen experimenteel geëvalueerd om uit te zoeken wat het beste taalmodel voor deze Nederlandstalige data is.

Hoofdstuk 2

Eenvoudige taalmodellering

In dit hoofdstuk komen de eenvoudige taalmodellen aan bod. In sectie 2.1 komen de meest gebruikte taalmodellen aan de beurt, namelijk N-grammen. Sectie 2.2 beschrijft cachingmodellen en de combinatie ervan met N-grammen. Tot slot wordt in sectie 2.3 de evaluatie van taalmodellen behandeld. Hier wordt het begrip perplexiteit geïntroduceerd.

2.1 *N*-grammen

De eenvoudigste en meest gebruikte taalmodellen zijn N-grammen. Allereerst zal de werking van N-grammen uitgelegd worden. In het volgende deel komt smoothing aan bod, een techniek om laagfrequente en nultellingen tegen te gaan. Vervolgens wordt een alternatieve methode besproken om met laagfrequente en nul tellingen om te gaan, namelijk backoff en interpolatie. Om te eindigen zal een overzicht gegeven worden van de voor- en nadelen van N-gram taalmodellen.

2.1.1 *N*-grammen als taalmodel

Stel dat aan de uitgang van een spraakherkenningssyteem volgend deel van een zin verschijnt:

De renner viel zwaar en werd verzorgd door de ...

Het is zeer waarschijnlijk dat het volgende woord *dokter* is. Het volgende woord in een zin is immers afhankelijk van de vorige woorden.

Het raden van het volgende woord hangt nauw samen met de kans dat een bepaalde opeenvolging van woorden voorkomt. Dat wilt zeggen dat algoritmes die kansen toewijzen aan zinnen ook kunnen gebruikt worden voor het toewijzen van kansen aan een volgend woord.
De N-gram taalmodellen maken hiervan gebruik om het volgende woord te voorspellen. Een bigram benadert de kans van een woord w_q op basis van het voorgaande woord:

$P(w_q|w_{q-1})$, een trigram op basis van de 2 voorgaande woorden: $P(w_q|w_{q-1}, w_{q-2})$. Algemeen, een N-gram benadert de kans van het volgende woord op basis van de $N-1$ vorige woorden: $P(w_q|w_{q-1}, w_{q-2}, ... \ w_{q-N+1})$.
In principe zou met de volledige geschiedenis rekening moeten worden gehouden, maar de bijdrage van de oudste woorden is veel kleiner dan die van de recentste woorden. Daarom wordt met slechts een beperkt deel van de geschiedenis rekening gehouden, namelijk de $N-1$ laatste woorden. Dit staat gekend als de Markov-veronderstelling:

$$P(w_q|w_1^{q-1}) \approx P(w_q|w_{q-N+1}^{q-1}) \tag{2.1}$$

waarbij w_j^{q-1} staat voor de geschiedenis van het j-de tot het $(q-1)$ste woord.

Opstellen model

Het N-gram model wordt opgesteld door het tellen van alle N-grammen die voorkomen in de trainingsset. Voor een bigram geldt dat de kans dat een woord voorkomt, gegeven het voorgaande woord, gelijk is aan het aantal keer $C(w_{q-1}w_q)$ dat dit specifieke bigram voorkomt in de trainingsdata gedeeld door het aantal bigrammen die beginnen met hetzelfde woord:

$$P(w_q|w_{q-1}) = \frac{C(w_{q-1}w_q)}{\sum_w C(w_{q-1}w)} \tag{2.2}$$

Het normaliseren is nodig opdat de som van de kansen gelijk is aan 1, anders is er geen sprake van kansen. Deze vergelijking kan nog vereenvoudigd worden, daar het aantal bigrammen dat begint met hetzelfde woord gelijk is aan het aantal keer dat dat woord voorkomt:

$$P(w_q|w_{q-1}) = \frac{C(w_{q-1}w_q)}{C(w_{q-1})} \tag{2.3}$$

Algemeen geldt dan voor een N-gram:

$$P(w_q|w_{q-N+1}^{q-1}) = \frac{C(w_{q-N+1}^{q-1}w_q)}{C(w_{q-N+1}^{q-1})} \tag{2.4}$$

Telkens wordt zo de relatieve frequentie van een N-gram gezocht voor het schatten van kansen. Het gaat hier om een maximum-likelihood-schatter, want de likelihood van de trainingsset T, gegeven het model M, $P(T|M)$ is maximaal.

2.1.2 Smoothing

Een groot probleem bij N-gram taalmodellen is dat ze moeten getraind worden op een beperkte trainingsset. Hierdoor is niet de volledige taal gemodelleerd en zullen niet alle mogelijke N-grammen in het model aanwezig zijn. Het is dan mogelijk dat bepaalde N-grammen een kans nul zullen krijgen. Stel dat het trigram *de Siberische tijger* niet aanwezig is in de trainingsset. Dan zal de kans $P(tijger|de, Siberische)$ gelijk zijn aan 0, wat niet realistisch is.
Dit gebrek wordt opgelost door het gebruiken van smoothingtechnieken. Deze technieken zullen een deel van de kansmassa van de hoogfrequente N-grammen toewijzen aan ongeziene en laagfrequente N-grammen.

Additieve smoothing

De eenvoudigste vorm van smoothing gebeurt door aan elk mogelijk N-gram een vast getal toe te voegen aan de telling ervan. Dit wordt additieve smoothing of ook wel *Laplace smoothing* genoemd. Een logische keuze voor het getal is 1. Dit heet add-one smoothing. Alle ongeziene N-grammen krijgen in dat geval de waarde 1 in de telling. De add-one smoothingtechniek geeft echter geen goede resultaten, zie [13], maar kan wel dienen als de basis voor betere technieken.

De nieuwe kansen kunnen als volgt berekend worden:

$$P_{add1}(w_q|w_{q-N+1}^{q-1}) = \frac{C(w_{q-N+1}^{q-1}w_q) + 1}{C(w_{q-N+1}^{q-1}) + |\mathcal{V}|} \tag{2.5}$$

waarbij $|\mathcal{V}|$ het aantal verschillende woorden in het taalmodel voorstelt, dit is de grootte van het vocabularium $\mathcal{V}$.
In de literatuur wordt smoothing dikwijls aanzien als discounting. Dit is omdat smoothing overeenkomt met het verlagen van de hoge kansen om een deel van de kansmassa toe te wijzen aan de laagfrequente tellingen. De discountfactor is logischerwijs gelijk aan:

$$d_C = \frac{C^*}{C} \tag{2.6}$$

met C^* de telling na discounting.

In plaats van 1 toe te voegen aan de telling van elk N-gram kan ook δ toegevoegd worden met $0 < \delta \leq 1$:

$$P_{add\delta}(w_q|w_{q-N+1}^{q-1}) = \frac{C(w_{q-N+1}^{q-1}w_q) + \delta}{C(w_{q-N+1}^{q-1}) + \delta\,|\mathcal{V}|} \tag{2.7}$$

De resultaten kunnen voor een goed gekozen δ beter zijn dan bij add-one smoothing, maar zijn nog steeds niet bevredigend, zie o.a. [5]. Het is duidelijk dat er betere smoothingtechnieken gewenst zijn.

Geavanceerde smoothingtechnieken

Andere bekende smoothingtechnieken zijn Witten-Bell discounting en Good-Turing discounting. Bij Witten-Bell discounting worden de kansen van ongeziene N-grammen gemodelleerd door de kansen van N-grammen die één keer voorkomen. Good-Turing discounting herberekent de hoeveelheid kansmassa voor N-grammen die weinig voorkomen door te kijken naar het aantal N-grammen met een hogere telling. In [18] worden deze technieken uitgebreider uitgelegd.

In [5] staan naast deze twee nog heel wat andere smoothingtechnieken beschreven. De absolute discountingmethode trekt een bepaalde waarde van de tellingen af en verdeelt deze kansmassa over de ongeziene N-grammen. De ongewijzigde Kneser-Ney smoothingtechniek gebruikt ook absolute discounting, maar hun backoff-kansen

(zie verder) zijn anders gedefinieerd. Bij de gewijzigde Kneser-Ney techniek wordt niet meer absolute discounting gebruikt, maar een variant hierop.
Uit [5] blijkt dat deze laatste twee smoothingtechnieken de beste resultaten geven.

2.1.3 Backoff en interpolatie

Een andere manier om met nultellingen om te gaan is het gebruiken van lagere orde tellingen. Als een bepaald trigram $w_{q-2}w_{q-1}w_q$ nul keer voorkomt, dan kan de kans van dit trigram geschat worden door te kijken naar de telling van het bigram $w_{q-1}w_q$. Als dit bigram ook niet voorkomt in het taalmodel, kan gekeken worden naar de kans van het unigram w_q. Het terugvallen op lagere orde tellingen gebeurt dus recursief. Nu zijn er twee mogelijkheden om dit te implementeren. Ofwel wordt er enkel teruggevallen op de lagere orde tellingen indien de telling van de standaardorde nul of laag is. Dit is de backoff methode. De andere mogelijkheid bestaat uit een gewogen som van de tellingen van zowel de standaardorde als de lagere ordes. Deze methode wordt interpolatie genoemd.

Backoff

Voor een trigram wordt de kans $\hat{P}(w_q|w_{q-2}w_{q-1})$ als volgt bepaald:

$$\hat{P}(w_q|w_{q-2}w_{q-1}) = \begin{cases} P(w_q|w_{q-2}w_{q-1}), & \text{als } C(w_{q-2}w_{q-1}w_q) > 0 \\ \alpha_1 P(w_q|w_{q-1}), & \text{als } C(w_{q-2}w_{q-1}w_q) = 0 \\ & \text{en } C(w_{q-1}w_q) > 0 \\ \alpha_2 P(w_q), & \text{anders.} \end{cases} \tag{2.8}$$

Als het trigram in kwestie telling nul heeft, wordt de kans vervangen door de kans van het overeenkomstige bigram, vermenigvuldigd met een backoffgewicht α. Het terugvallen op lagere ordes gebeurt zoals eerder gezegd recursief.

Algemeen geldt:

$$\hat{P}(w_q|w_{q-N+1}^{q-1}) = \tilde{P}(w_q|w_{q-N+1}^{q-1}) + \theta(P(w_q|w_{q-N+1}^{q-1}))\alpha\hat{P}(w_q|w_{q-N+2}^{q-1}) \tag{2.9}$$

hierbij is $\theta(x)$ een functie die de waarde 1 aanneemt als het argument nul is en voor de rest nul is.

Om ervoor te zorgen dat er met echte kansen blijft gewerkt worden, moet een deel van de kansmassa van de hoogste orde tellingen verdeeld worden over de lagere orde tellingen. Hiervoor is discounting noodzakelijk. Dat is de reden dat in formule (2.9) $\tilde{P}$ wordt gebruikt en niet P. De $\tilde{P}$ komt overeen met de kans P na discounting:

$$\tilde{P}(w_q|w_{q-N+1}^{q-1}) = \frac{C^*(w_{q-N+1}^{q})}{C(w_{q-N+1}^{q-1})} \tag{2.10}$$

Ten slotte moeten ook nog de α-waardes bepaald worden. Stel dat de overgebleven kansmassa voor de lagere ordes wordt voorgesteld door de functie β:

$$\beta(w_{q-N+1}^{q-1}) = 1 - \sum_{w_q : C(w_{q-N+1}^{q})>0} \tilde{P}(w_q|w_{q-N+1}^{q-1}) \tag{2.11}$$

Dit is de totale kansmassa die verdeeld mag worden over de $(N-1)$-grammen. Om nu een deel van deze kansmassa toe te wijzen aan een individueel $(N-1)$-gram, is er normalisatie van β nodig:

$$\alpha(w_{q-N+1}^{q-1}) = \frac{1 - \sum_{w_q : C(w_{q-N+1}^{q})>0} \tilde{P}(w_q|w_{q-N+1}^{q-1})}{1 - \sum_{w_q : C(w_{q-N+1}^{q})>0} \tilde{P}(w_q|w_{q-N+2}^{q-1})} \tag{2.12}$$

De α-waarde hangt af van de vorige $N-1$ woorden, maar is onafhankelijk van het huidige woord.

Samengevat geeft dit voor een trigram het volgende backoffmodel:

$$\hat{P}(w_q|w_{q-2}w_{q-1}) = \begin{cases} \tilde{P}(w_q|w_{q-2}w_{q-1}), & \text{als } C(w_{q-2}w_{q-1}w_q) > 0 \\ \alpha(w_{q-2}^{q-1})\tilde{P}(w_q|w_{q-1}), & \text{als } C(w_{q-2}w_{q-1}w_q) = 0 \\ & \text{en } C(w_{q-1}w_q) > 0 \\ \alpha(w_{q-1})\tilde{P}(w_q), & \text{anders.} \end{cases} \tag{2.13}$$

Interpolatie

Bij de backoff methode werd slechts gebruik gemaakt van lagere orde telling indien er een nultelling was. Interpolatiemethodes gebruiken echter altijd lagere orde tellingen bij het berekenen van kansen. Voor een trigram geldt voor de kans $\hat{P}(w_q|w_{q-2}w_{q-1})$:

$$\hat{P}(w_q|w_{q-2}w_{q-1}) = \lambda_1 P(w_q|w_{q-2}w_{q-1}) + \lambda_2 P(w_q|w_{q-1}) + \lambda_3 P(w_q) \tag{2.14}$$

met

$$\sum_i \lambda_i = 1 \tag{2.15}$$

Net als bij backoff kunnen de parameters λ_i afhankelijk gemaakt worden van de voorgaande woorden:

$$\hat{P}(w_q|w_{q-2}w_{q-1}) = \lambda_1(w_{n-2}^{n-1})P(w_q|w_{q-2}w_{q-1}) + \lambda_2(w_{n-2}^{n-1})P(w_q|w_{q-1}) + \lambda_3(w_{n-2}^{n-1})P(w_q) \tag{2.16}$$

De λ-waarden kunnen getraind worden door de likelihood op een stuk tekst te maximaliseren. Het stuk tekst mag echter niet gebruikt zijn om het N-gram model op te stellen. Dit wordt *held out interpolation* genoemd. Een andere manier bestaat erin telkens verschillende stukjes tekst uit de gebruikte trainingsset te halen, hiermee de λ-waarden te trainen en vervolgens de resultaten uit te middelen. Dit is *deleted interpolation*.

2.1.4 Discussie

Het gebruik van N-gram taalmodellen biedt zowel voor- als nadelen. Beiden worden in deze sectie t.o.v. elkaar afgewogen. Om te beginnen wordt een overzicht gegeven van de sterke punten van N-grammen:

- Een groot voordeel van N-gram taalmodellen is dat er geen kennis van grammatica vereist is. Dit wilt zeggen dat de structuur van de modellen onafhankelijk is van de gebruikte taal. Het enige dat zal verschillen tussen talen onderling is het gebruikte vocabularium.
- Een ander sterk punt van N-gram taalmodellen is dat het heel goed lokale afhankelijkheden kan modelleren. Deze afhankelijkheden zitten meestal vervat in de $N-1$ vorige woorden die relevant zijn in het N-gram taalmodel.

  Voorbeeld 2.1 Een 3-gram taalmodel geeft volgende kans terug

  $$P(York|in, New) = 0.91$$

  De lokale beperking is hier dat na *in New* meestal het tweede deel *York* van de miljoenenstad *New York* voorkomt. Het N-gram taalmodel modelleert deze beperking zeer goed, getuige de grote kans die het aan dit trigram geeft.

Naast deze voordelen bieden N-gram taalmodellen echter ook enkele nadelen:

- Van zodra echter een afhankelijkheid over meer dan N woorden uitgestrekt is, kan het niet meer gemodelleerd worden door een N-gram taalmodel. Dit probleem wordt geïllustreerd in volgend voorbeeld.

  Voorbeeld 2.2 Beschouw een 3-gram taalmodel en volgende zin:

  Gaston en Leo hebben een optreden in het sportpaleis gegeven.

  $$\Rightarrow P(Leo|Gaston, en) = 0.79 \tag{2.17}$$

  Wanneer er echter meer dan 1 woord tussen *Gaston* en *Leo* staat, loopt het mis:

  Gaston, die maar net op tijd was, en Leo hebben een optreden in het sportpaleis gegeven.

  $$\Rightarrow P(Leo|was, en) = 0.00002 \tag{2.18}$$

- Een ander minpunt van N-gram taalmodellen is de sparsiteit van de data. Heel wat N-grammen komen niet voor in de trainingsdata en deze worden dan benaderd via smoothing en backoff/interpolatie. Deze benaderingen zijn echter dikwijls niet betrouwbaar. Neem bijvoorbeeld een 5-gram. Als er 30000 woorden in het vocabularium zijn, zijn er $(30000)^5 \approx 10^{22}$ mogelijke 5-grammen. Om elk 5-gram de kans te geven minstens 1 keer voor te komen, is er al een trainingsset nodig met minstens 10^{22} woorden. Dit is zelfs met de huidige hoeveelheid beschikbare data lang niet mogelijk. Natuurlijk zullen er ook 5-grammen zijn die nooit zullen voorkomen, zoals *ik jij het boom naast*, maar zelfs voor de overige 5-grammen zal er niet genoeg data zijn.

- Ten slotte volgt nog een laatste, niet onbelangrijke zwakte van N-gram taalmodellen. Op geen enkele manier houden ze rekening met de betekenis van woorden. Zo is voor een N-gram taalmodel *De renner werd verzorgd door de dokter* helemaal verschillend van *De renner werd verzorgd door de arts*. Dit is echter niet gewenst. Om semantische overeenkomsten tussen woorden in rekening te brengen, zijn er geavanceerdere modellen nodig. Voorbeelden hiervan zijn LSA (hoofdstuk 3) en pLSA (hoofdstuk 4).

2.2 Cachingmodellen

Een andere eenvoudige vorm van taalmodellen zijn cachingmodellen. Eerst wordt het principe hiervan uitgelegd en vervolgens wordt de werking geïllustreerd. Tot slot wordt de combinatie van cachingmodellen met N-grammen uitgelegd.

2.2.1 Principe

Een minpunt van een N-gram taalmodel is dat het enkel rekening houdt met de $N-1$ vorige woorden. Meestal wordt N beperkt gehouden tot maximaal 3. Dit komt erop neer dat er bij het voorspellen van het volgende woord geen rekening wordt gehouden met alle woorden die meer dan 2 plaatsen voor dit woord staan. Dit is een te grote beperking. Er bestaat immers een redelijke kans dat het volgende woord reeds voorgekomen is in een recente geschiedenis. Cachingmodellen [19] steunen op dit principe.

Voorbeeld 2.3 Stel dat volgende tekst voorkomt:

Philippe Gilbert heeft gisteren op een indrukwekkende wijze de zevende tourrit gewonnen. In een sprint met 4 was Gilbert duidelijk de snelste. Op 15 km van de meet was Gilbert nochtans betrokken geraakt in een valpartij.

Door de aanwezigheid van het woord *Gilbert* in de eerste zin, is de kans groot dat dit woord nog zal terugkomen in één van de volgende zinnen. Cachingmodellen houden hier rekening mee en zullen een vrij grote kans geven aan *Gilbert* om in de tweede zin voor te komen. Nadat *Gilbert* nog eens in de tweede zin is voorgekomen, zal de kans op dit woord nog meer stijgen. In de derde zin zal het model een nog hogere kans geven aan *Gilbert* om voor te komen.

2.2.2 Uitwerking

Er kan nu verder gewerkt worden richting het bepalen van kansen. Laat $h = w_{q-K}, ..., w_{q-1}$ de meest recente geschiedenis van w_q zijn. Dit wordt ook wel de cache genoemd. De eenvoudigste manier om tot een kans te komen is simpelweg te kijken naar het aantal keer $C_{\text{cache}}(w_q)$ dat een bepaald woord w_q voorkomt in de cache. De kans op optreden van dit woord wordt dan bekomen door dit aantal te delen door de

grootte K van de cache [6]:

$$P_{cache}(w_q|w_{q-K}, w_{q-K+1}, ..., w_{q-1}) = \frac{C_{\text{cache}}(w_q)}{K} \tag{2.19}$$

De grote beperking van deze uniforme cache is dat elk woord in de geschiedenis een even groot aandeel heeft in de kansbepaling. Beter zou zijn om de afhankelijkheid van de meest recente woorden te verhogen t.o.v. minder recente woorden. Dit kan bekomen worden door het aandeel van een woord exponentieel te laten dalen naarmate er verder in de geschiedenis wordt teruggegaan, dit wordt een exponentiële cache genoemd [6]:

$$P_{cache}(w_q|w_1, w_2, ..., w_{q-1}) = \beta \sum_{j=1}^{q-1} I(w_q = w_j) e^{-\alpha(q-j)} \tag{2.20}$$

hierbij is I een functie zodanig dat $I(x) = 1$ wanneer x waar is en $I(x) = 0$ anders, α is de afnamefactor en β is een normalisatiefactor. De afnamefactor α kan experimenteel bepaald worden via een validatieset.

De beschreven modellen zijn allemaal 1-gram cachingmodellen, omdat slechts gekeken wordt of enkel het huidige woord in de cache zit. Hogere orde cache modellen bestaan ook. In [28] staat beschreven hoe deze berekend kunnen worden. Er wordt hier verder niet op ingegaan.

2.2.3 Combinatie met N-grammen

Cachingmodellen op zich zijn onvoldoende betrouwbaar als taalmodel. Door de gelimiteerde cachegrootte zijn de modellen zeer spaars. Een oplossing hiervoor is ze te combineren met N-grammen. De eenvoudigste methode hiervoor is lineaire interpolatie [6]:

$$P(w_q|w_1, ..., w_{q-1}) = (1-\lambda) P_{N\text{-gram}}(w_q|w_{q-N+1}, ..., w_{q-1}) + \lambda P_{\text{cache}}(w_q|w_1, ..., w_{q-1}) \tag{2.21}$$

De interpolatieparameter λ ($0 \leq \lambda \leq 1$) kan experimenteel bepaald worden via een validatieset.

2.3 Evaluatie taalmodel

Als maatstaf voor de evaluatie van een taalmodel wordt dikwijls perplexiteit gebruikt. Een andere veel gebruikte maat is de word-error rate (WER), dit is het percentage fout herkende woorden. Het nadeel van deze maat is dat er een volledig spraakherkenningssysteem vereist is om deze te berekenen. De perplexiteit kan daarentegen berekend worden uit enkel het taalmodel. In deze masterproef zal perplexiteit gebruikt worden om taalmodellen te evalueren. Om dit begrip te kunnen begrijpen wordt eerst een ander begrip, entropie, ingeleid.

2.3.1 Entropie

Entropie is een maat voor de onzekerheid verbonden aan een random variabele. Wiskundig wordt de entropie H van een discrete random variabele X met mogelijke waardes $\{x_1, x_2, ..., x_n\}$ en kansdichtheidsfunctie $p(x)$ als volgt gedefinieerd:

$$H(X) = -\sum_{i=1}^{n} p(x_i) \log_2 p(x_i) \tag{2.22}$$

Hierbij is $p(x) \log_2 p(x) = 0$ als $p(x) = 0$. Als grondtal van het logaritme kan eerder welke waarde gekozen worden. Omdat hier als grondtal 2 gekozen is, wordt de entropie uitgedrukt in bits.

Het begrip entropie kan geïllustreerd worden vanuit het standpunt van codering. Stel dat er een aantal symbolen, met gekende kansen, moeten gecodeerd worden. Dan is de entropie van de symbolen een ondergrens voor het gemiddelde aantal bits per symbool dat nodig is om een boodschap te coderen.
Dit wordt duidelijk gemaakt aan de hand van een voorbeeld.

Voorbeeld 2.4 Stel dat er 4 symbolen A, B, C en D moeten gecodeerd worden. In het eerste geval heeft elk symbool een even grote kans om op te treden, dus $P(A) = P(B) = P(C) = P(D) = 1/4$. Dan is de entropie gelijk aan:

$$H(X) = -\sum_{i=1}^{4} p(x_i) \log_2 p(x_i) = -4.\frac{1}{4} \log_2(1/4) = 2 \text{ bits} \tag{2.23}$$

Gemiddeld zullen er bijgevolg 2 bits/symbool nodig zijn om een boodschap te coderen.

Stel nu echter dat één van de symbolen een veel grotere kans dan de anderen heeft om op te treden. Neem bv. $P(A) = 3/4, P(B) = P(C) = P(D) = 1/12$. Dan is de entropie veranderd:

$$\begin{aligned} H(X) &= -\sum_{i=1}^{4} p(x_i) \log_2 p(x_i) \\ &= -\frac{3}{4}.\log_2(\frac{3}{4}) - \frac{1}{12}.\log_2(\frac{1}{12}) - \frac{1}{12}.\log_2(\frac{1}{12}) - \frac{1}{12}.\log_2(\frac{1}{12}) \\ &= 1,21 \text{ bits} \end{aligned} \tag{2.24}$$

Aangezien één van de symbolen een veel grotere kans heeft om op te treden dan de anderen, is de onzekerheid over de symbolen gezakt en ligt de entropie dus lager. Er zijn nu gemiddeld minder bits nodig om een symbool te coderen.

De perplexiteit wordt gedefinieerd als 2^H. Uit de definitie volgt dat de perplexiteit gelijk is aan het gewogen gemiddelde aantal keuzes waaruit een variabele moet kiezen. Anders gezegd geeft de perplexiteit aan dat het raden van de waarde van de variabele X even moeilijk is als het kiezen uit een aantal mogelijkheden die optreden met een zelfde waarschijnlijkheid. Dit aantal is gelijk aan de perplexiteit.

Toegepast op het voorbeeld van codering geeft dit voor het eerste geval een perplexiteit van 4. Dit is logisch, aangezien de 4 symbolen met een zelfde waarschijnlijkheid voorkomen. In het tweede geval is de perplexiteit echter 2,31. Het raden van de juiste X is nu even moeilijk als het kiezen uit 2,31 mogelijkheden met een zelfde waarschijnlijkheid.

2.3.2 Entropie per woord

Entropie kan gebruikt worden als een maatstaf voor hoe goed een grammatica een taal benadert, ofwel hoe voorspellend een gegeven grammatica is met betrekking tot het volgende woord in een zin. Voorlopig is slechts de entropie van één enkele variabele gedefinieerd. Voor taalmodellen is echter de entropie van een reeks van woorden $w_1 w_2 ... w_n = w_1^n$ gewenst.
De oplossing bestaat erin een variabele te hebben, die als bereik een eindige reeks van woorden heeft. De entropie van zo'n variabele kan als volgt berekend worden [18]:

$$H(w_1^n) = -\sum_{w_1^n \in L} p(w_1^n) \log p(w_1^n) \tag{2.25}$$

met L een stochastisch proces.

De entropie per woord is gelijk aan de entropie van een reeks woorden gedeeld door het aantal woorden:

$$\frac{1}{n} H(w_1^n) = -\frac{1}{n} \sum_{w_1^n \in L} p(w_1^n) \log p(w_1^n) \tag{2.26}$$

Om de entropie van een taal te berekenen, moet een oneindige reeks van woorden beschouwd worden. Als wordt aangenomen dat een taal een stochastisch proces L is dat een reeks van woorden genereert, wordt haar entropie $H(L)$ gedefinieerd als:

$$\begin{aligned} H(L) &= \lim_{n\to\infty} \frac{1}{n} H(w_1^n) \\ &= \lim_{n\to\infty} \frac{1}{n} \sum_{w_1^n \in L} p(w_1^n) \log p(w_1^n) \end{aligned} \tag{2.27}$$

Het theorema van Shannon-McMillan-Breiman stelt dat als de taal stationair en ergodisch is [1],

$$H(L) = \lim_{n\to\infty} -\frac{1}{n} \log p(w_1^n) \tag{2.28}$$

Eén enkele reeks van woorden die lang genoeg is, is voldoende in plaats van te sommeren over alle mogelijke reeksen.

Samengevat komt het erop neer dat de entropie van een stochastisch proces kan gevonden worden door een zeer lange reeks output te nemen. De entropie is dan gelijk aan de gemiddelde log probabiliteit van deze reeks. Dit geldt echter alleen onder bepaalde voorwaarden, namelijk het stationair en ergodisch zijn van de taal. Hieraan is echter niet altijd voldaan.

2.3.3 Cross-entropie

Hoe kan een taalmodel nu geëvalueerd worden? Daarvoor wordt het begrip cross-entropie geïntroduceerd. Cross-entropie is nuttig wanneer de werkelijke probabiliteitsverdeling p onbekend is. In natuurlijke taal is enkel een benadering van p, nl. m, gekend. De cross-entropie van m over p is gedefinieerd als [18]:

$$H(p,m) = \lim_{n\to\infty} \frac{1}{n} \sum_{W\in L} p(w_1, w_2, ..., w_n) \log m(w_1, w_2, ..., w_n) \tag{2.29}$$

Volgens het theorema van Shannon-McMillan-Breiman volgt nu voor een stationair, ergodisch proces:

$$H(p,m) = \lim_{n\to\infty} -\frac{1}{n} \log m(w_1 w_2 ... w_n) \tag{2.30}$$

Een interessante eigenschap zegt dat de cross-entropie $H(p,m)$ een bovengrens is voor de werkelijke entropie $H(p)$:

$$H(p,m) \geq H(p) \tag{2.31}$$

Hieruit volgt dat de cross-entropie van een taalmodel een goede maat is voor de kwaliteit ervan. Hoe lager de cross-entropie, hoe dichter de werkelijke entropie benaderd wordt en hoe nauwkeuriger het model is. Twee modellen m_1 en m_2 kunnen nu met elkaar vergeleken worden. Het model met de laagste cross-entropie zal het nauwkeurigste zijn. De kwaliteit van een taalmodel zal echter steeds uitgedrukt worden in termen van perplexiteiten, maar dit komt op hetzelfde neer aangezien de perplexiteit gelijk is aan 2^H.

2.4 Besluit

Eenvoudige taalmodellen zoals N-grammen en cachingmodellen zijn een goed vertrekpunt voor het modelleren van natuurlijke taal. N-grammen modelleren goed de lokale afhankelijkheden binnen een taal, terwijl cachingmodellen rekening houden met recent gebruikte woorden. Beide modellen houden echter geen rekening met de betekenis van woorden. Hiervoor zijn geavanceerdere modellen nodig. Deze modellen vormen het onderwerp van de volgende twee hoofdstukken.
Verder werd het begrip perplexiteit ingeleid, dat zal gebruikt worden bij de evaluatie van een taalmodel.

Hoofdstuk 3

Latent semantische analyse

3.1 Inleiding

Eenvoudige taalmodellen als N-grammen en cachingmodellen bleken niet in staat om rekening te houden met onderlinge betekenisovereenkomsten tussen woorden. Daarnaast bleek het beperkte afhankelijkheidsgebied van N-gram taalmodellen een probleem. Herneem nu voorbeeld 2.2. Terwijl het model erin slaagde het woord *Leo* in de eerste zin goed te modelleren, deed het het slecht bij dit woord in de tweede zin. De reden is dat N-gram taalmodellen enkel en alleen rekening houden met de $N-1$ vorige woorden. Een betere oplossing in dit geval zou zijn door te kijken naar alle voorgaande woorden in de zin. Dan valt op dat het woord *Gaston* reeds voorgekomen is, en aangezien *Gaston* en *Leo* dikwijls samen voorkomen in een tekst, zou de kans op het woord *Leo* in deze zin vrij hoog moeten zijn.
Het komt erop neer dat woorden die een gelijkaardige betekenis hebben dikwijls in dezelfde teksten voorkomen. Latent semantische analyse (LSA) gaat zelfs nog een stap verder door er van uit te gaan dat woorden die betekenisverwant zijn, in gelijkaardige contexten voorkomen [27]. LSA slaagt erin om als het ware verborgen betekenisovereenkomsten tussen woorden te ontdekken. Vandaar komt ook de naam latent (=verborgen) semantische analyse.

Om LSA optimaal te kunnen toepassen, is het gewenst dat de trainingsset of het corpus $\mathcal{T}$ in afzonderlijke documenten is opgesplitst. LSA stelt hieruit een frequentiematrix op en berekent van deze matrix een singuliere waarden ontbinding. Na een dimensionaliteitsreductie vormt elk woord en document een punt in een hoogdimensionale ruimte. Op basis van afstanden tussen woorden en documenten in deze ruimte kan een kans worden afgeleid voor het volgende woord. In plaats van enkel rekening te houden met de $N-1$ vorige woorden, voorspelt LSA het volgende woord op basis van alle woorden voorafgaand in hetzelfde document.

Om te beginnen wordt in sectie 3.2 besproken hoe een LSA-model opgebouwd wordt. In sectie 3.3 worden enkele belangrijke toepassingen van LSA gegeven. Vervolgens komt in sectie 3.4 specifiek de toepassing van LSA binnen spraakherkenning

aan bod. In sectie 3.5 wordt een overzicht gegeven van de sterktes en zwaktes van LSA. Dan wordt in sectie 3.6 de combinatie met N-grammen besproken. In sectie 3.7 wordt besproken hoe N-grammen rechtstreeks in de frequentiematrix van LSA kunnen ingevoerd worden.

3.2 Opbouw

3.2.1 Opstellen van de frequentiematrix

De eerste stap bestaat erin de frequentiematrix W op te stellen. W wordt ook wel de term-document matrix genoemd. Er wordt verondersteld dat het corpus $\mathcal{T}$ onderverdeeld is in documenten of artikels. Elke rij in de frequentiematrix W stelt een woord in het vocabularium $\mathcal{V}$ voor, met $|\mathcal{V}| = M$. Elke kolom in de matrix W komt overeen met een document. In totaal zijn er N documenten. De waarde van een element c_{ij} van de matrix W is gelijk aan het aantal keer dat woord w_i in document d_j voorkomt. W is dus een $(M \times N)$ matrix. Aangezien documenten meestal maar een beperkt aantal woorden bevatten, zal de matrix W zeer spaars zijn.
Het doel is nu om een mapping te vinden tussen de sets $\mathcal{V}$, $\mathcal{T}$, en een vectorruimte $\mathcal{S}$ van dimensie k, waarbij elk woord uit $\mathcal{V}$ en elk document uit $\mathcal{T}$ voorgesteld wordt door een vector in $\mathcal{S}$. Een belangrijke eigenschap van deze ruimte is dat twee woorden wiens voorstellingen in deze ruimte, volgens een geschikte metriek, dicht bij elkaar liggen, de neiging hebben om in hetzelfde soort documenten voor te komen. Het maakt niet uit of ze nu werkelijk binnen dezelfde documenten zijn voorgekomen of niet. Omgekeerd geldt ook dat twee documenten wiens voorstellingen dicht bij elkaar liggen in deze ruimte, de neiging hebben om dezelfde betekenis in te houden, of ze nu dezelfde woorden bevatten of niet. Hieruit volgt dat de voorstellingen van respectievelijk de termen en de documenten, die semantisch verwant zijn, ook dicht bij elkaar in deze ruimte zullen liggen [3].

3.2.2 Wegen van de elementen

Een niet onbelangrijke voorbereidingsstap is het wegen van de elementen uit de frequentiematrix W. Het idee van wegen is om meer gewicht aan verrassende gebeurtenissen te geven dan aan verwachte gebeurtenissen [27]. In [2] wordt een onderscheid gemaakt tussen functiewoorden, die in haast elke tekst voorkomen, bv. *de*, *op*, en inhoudswoorden, dit zijn woorden die minder voorkomen en veel meer informatie bevatten, bv. *beer* of *Gaston*. Omdat functiewoorden minder informatie bevatten, moeten deze een lager gewicht krijgen dan inhoudswoorden.
De weging van de elementen gebeurt steeds in twee stappen: een lokale weging $L(i, j)$ van een term i in een document j en een globale weging $G(i)$ van een term i in heel het corpus. De gewogen termfrequentie W_{ij} is dan gelijk aan:

$$W_{ij} = G_i L_{ij} \tag{3.1}$$

Als lokale weging L_{ij} wordt dikwijls het volgende gedaan [2] [10]:

$$L_{ij} = \log_2(1 + c_{ij}) \tag{3.2}$$

Het logaritme zorgt voor een demping van de grote frequenties binnen één document.

Twee woorden die evenveel voorkomen in hetzelfde document, dragen daarom niet altijd evenveel bij tot de inhoud van het document. Hun bijdrage hangt ook af van hoeveel keer ze in heel het corpus voorkomen. Globale weging zorgt ervoor dat deze elementen een verschillende weging krijgen. Er zijn verschillende methoden voor het uitvoeren van een globale weging, zie bv. [10]. De populairste methode is de entropieweging, gebruikt in o.a. [3]. Deze methode werkt als volgt:
Bereken eerst de relatieve frequentie f_{ij} van de term w_i binnen het document d_j:

$$f_{ij} = \frac{c_{ij}}{t_i} \tag{3.3}$$

hierbij is t_i het aantal keer dat w_i voorkomt in heel het corpus.

De genormaliseerde entropie E_i van w_i is gelijk aan:

$$E_i = -\frac{1}{\log_2(N)} \sum_{j=1}^{N} f_{ij} \log_2 f_{ij} \tag{3.4}$$

Uit de definitie volgt dat $0 \leq E_i \leq 1$. Hoe hoger de waarde van E_i, des te meer de term w_i verspreid is over de documenten en des te kleiner zijn informatieve waarde. Omgekeerd als E_i klein is, dan komt de term w_i voor in weinig documenten en heeft het een grote informatieve waarde. Het definiëren van de globale weging G_i is nu triviaal:

$$G_i = 1 - E_i \tag{3.5}$$

3.2.3 Singuliere waarden ontbinding

De volgende stap is het berekenen van de singuliere waarden ontbinding (SWO) van de frequentiematrix W:

$$W = USV^T \tag{3.6}$$

waarbij U een $(M \times M)$ vierkante matrix is, S is een $(M \times N)$ rechthoekige diagonale matrix en V is een $(N \times N)$ vierkante matrix. Per definitie is S positief definiet en zijn zowel U als V unitair, dus $U^TU = I$ en $V^TV = I$.

Vervolgens zal er een lage rang benadering $\hat{W}$ van deze matrix gezocht geworden. Deze benadering komt tot stand door enkel de k ($k \ll \min(M, N)$) grootste singuliere waarden te behouden en de andere waarden op 0 te zetten:

$$\hat{W} = U_k S_k V_k^T \tag{3.7}$$

dit wordt ook wel de afgebroken SWO van W genoemd. Nu is U_k een $(M \times k)$ matrix van linker singuliere vectoren u_i, S_k is een $(k \times k)$ diagonale matrix en V_k is een $(N \times k)$ matrix van rechter singuliere vectoren v_j. $\hat{W}$ is de matrix van rang k die het best de oorspronkelijke matrix benadert met betrekking tot de Frobeniusnorm. Dus $\hat{W}$ minimaliseert $\left\|\hat{W} - W\right\|_F$ over alle matrices $\hat{W}$ van rang k.
Schematisch ziet deze benaderde ontbinding er als volgt uit:

$$\underset{W}{\begin{bmatrix} c_{1,1} & \dots & c_{1,n} \\ \vdots & \ddots & \vdots \\ c_{m,1} & \dots & c_{m,n} \end{bmatrix}} \approx (\hat{\mathbf{w}}_i^T) \rightarrow \underset{U_k}{\begin{bmatrix} \begin{bmatrix} & \mathbf{u}_1 & \end{bmatrix} \\ \vdots \\ \begin{bmatrix} & \mathbf{u}_m & \end{bmatrix} \end{bmatrix}} \underset{S_k}{\begin{bmatrix} s_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & s_k \end{bmatrix}} \underset{V_k^T}{\begin{bmatrix} \begin{bmatrix} \\ \mathbf{v}_1 \\ \\ \end{bmatrix} & \dots & \begin{bmatrix} \\ \mathbf{v}_n \\ \\ \end{bmatrix} \end{bmatrix}} \uparrow (\hat{\mathbf{d}}_j) \tag{3.8}$$

Deze lage rangdecompositie biedt verschillende voordelen:

- Daar waar de oorspronkelijke matrix W zeer spaars was, is de gereduceerde matrix $\hat{W}$ dit niet meer. Sparsiteit kan gezien worden als een tekort aan data. De gereduceerde matrix $\hat{W}$ is een manier om de ontbrekende data te simuleren [27].
- Een tweede voordeel is dat de decompositie een vectorruimte van lage dimensie k definieert, namelijk de ruimte gespannen door zowel de linker als rechter singuliere vectoren. De i-de linker singuliere vector u_i is een voorstelling van term w_i in deze ruimte. Anderzijds is de j-de rechter singuliere vector v_j een voorstelling van document d_j in deze ruimte, zie ook vergelijking 3.8. Deze ruimte van dimensie k is dus de gezochte ruimte $\mathcal{S}$.
- Een laatste voordeel staat ook in [27] beschreven. De afgebroken SWO zorgt voor een reductie van ruis. De benadering $\hat{W}$ kan gezien worden als een smoothere versie van de originele matrix W. Als de matrix W samengesteld is uit een mengeling van signaal en ruis, met meer signaal dan ruis, dan zal het behouden deel $U_k S_k V_k^T$ de variatie in W door het signaal omvatten. De niet weerhouden vectoren in USV^T omvatten voornamelijk de variatie veroorzaakt door de ruis.

3.2.4 Afstand tussen woorden en documenten

In de nieuwe ruimte $\mathcal{S}$ is elk woord w_i voorgesteld door een linker singuliere vector u_i van dimensie k, en is elk document d_j voorgesteld door een rechter singuliere vector v_j, eveneens van dimensie k. Dit biedt de mogelijkheid om een afstand tussen woorden onderling, documenten onderling en tussen een woord en een document te definiëren.
In de matrix W zitten alle structurele verbanden tussen woorden en documenten vervat. De matrix WW^T karakteriseert alle co-occurrenties tussen woorden en de matrix W^TW karakteriseert alle co-occurrenties tussen documenten. De mate waarin twee woorden u_i en u_j een gelijkaardig voorkomen over de documenten hebben, kan

dus worden afgeleid uit het (i, j)-de element van WW^T. Anderzijds kan de mate waarin twee documenten v_i en v_j een gelijkaardig patroon van woorden bevatten, afgeleid worden uit het (i, j)-de element van W^TW.

Alvorens verder richting een afstand tussen woorden en documenten te gaan, wordt eerst het begrip cosinus van twee vectoren ingeleid.
Laat x en y twee vectoren zijn van lengte n:

$$\mathbf{x} = \langle x_1, x_2, ..., x_n \rangle \tag{3.9}$$
$$\mathbf{y} = \langle y_1, y_2, ..., y_n \rangle \tag{3.10}$$

De cosinus van de hoek θ tussen $\mathbf{x}$ en $\mathbf{y}$ is gelijk aan:

$$\cos(\mathbf{x}, \mathbf{y}) = \frac{\sum_{i=1}^{n} x_i \cdot y_i}{\sqrt{\sum_{i=1}^{n} x_i^2 \sum_{i=1}^{n} y_i^2}} \tag{3.11}$$
$$= \frac{\mathbf{x} \cdot \mathbf{y}}{\sqrt{\mathbf{x} \cdot \mathbf{x}} \cdot \sqrt{\mathbf{y} \cdot \mathbf{y}}} \tag{3.12}$$
$$= \frac{\mathbf{x}}{\|\mathbf{x}\|} \cdot \frac{\mathbf{y}}{\|\mathbf{y}\|} \tag{3.13}$$

De cosinus van de hoek tussen twee vectoren is dus niets anders dan het inwendig product van de vectoren, nadat deze genormaliseerd zijn. De cosinus varieert van -1 als beide vectoren in tegengestelde richting liggen ($\theta = 180°$) tot $+1$ wanneer beide vectoren in dezelfde richting liggen ($\theta = 0°$). Als de vectoren orthogonaal zijn ($\theta = 90°$), dan is de cosinus gelijk aan 0.
Een kenmerk van de cosinus tussen twee vectoren is dat deze enkel een maat is voor de richting van de vectoren en onafhankelijk is van hun lengte. Dit kan in bepaalde situaties, waar lengtes niet van belang zijn, van groot nut zijn. Een voorbeeld hiervan is het berekenen van de mate van semantische overeenkomst tussen woorden en documenten, zie verder in deze sectie.

Na dit intermezzo kan er verder gewerkt worden richting een afstand tussen woorden en documenten. Omwille van notatieredenen wordt in het vervolg van de masterproef met de matrix USV steeds de benadering $U_kS_kV_k$ van W bedoeld. De matrix WW^T kan dan volgens vergelijking 3.7 ook benaderend geschreven worden als:

$$WW^T \approx USV^TVS^TU^T \tag{3.14}$$
$$= US^2U^T \tag{3.15}$$

aangezien $V^T = V^{-1}$ en $S = S^T$. Omdat S diagonaal is, kan het (i, j)-de element van WW^T verkregen worden uit het inwendig product van de i-de en j-de rij van de matrix US, namelijk u_iS en u_jS. Anders gezegd, hoe dicht u_i ligt bij u_j in de ruimte $\mathcal{S}$ wordt gevonden uit het inwendig product tussen u_iS en u_jS. Als beide vectoren nog gedeeld worden door hun lengte, kan er een metriek beschouwd worden, die aangeeft hoe dicht u_i bij u_j ligt:

$$K(u_i, u_j) = \frac{u_iS^2u_j^T}{\|u_iS\| \, \|u_jS\|} \tag{3.16}$$

Dit is niets anders dan de cosinus tussen de twee vectoren $u_i S$ en $u_j S$:

$$K(u_i, u_j) = \cos(u_i S, u_j S) \tag{3.17}$$

Een waarde van $K(u_i, u_j)$ dicht bij 1 wil zeggen dat twee woorden altijd in gelijkaardige contexten voorkomen en bijgevolg semantisch verwant zijn. Een waarde van $K(u_i, u_j)$ dicht bij -1 duidt erop dat de twee woorden in steeds verschillende contexten voorkomen en semantisch volledig verschillen. Omdat de waarde van de metriek ook negatief kan zijn, definieert het geen afstand. Maar via een eenvoudige transformatie kan dit wel omgevormd worden tot een afstand [3].
Op een analoge manier kan een afstand tussen documenten worden afgeleid. De matrix $W^T W$ kan benaderend geschreven worden als:

$$\begin{aligned} W^T W &\approx V S^T U^T U S V^T && (3.18) \\ &= V S^2 V^T && (3.19) \end{aligned}$$

Een maat voor semantisch verwantschap tussen twee documenten v_i en v_j is nu:

$$K(v_i, v_j) = \frac{v_i S^2 v_j^T}{\|v_i S\| \, \|v_j S\|} = \cos(v_i S, v_j S) \tag{3.20}$$

Naast de cosinus bestaan er nog andere methodes om de afstand tussen woorden of documenten te berekenen. Enkele voorbeelden zijn Euclidische, Manhattan, Hellinger, Bhattacharya en Kullback-Leibler afstand [27]. In [4] zijn deze samen met de cosinus met elkaar vergeleken. Het ging telkens om tests i.v.m. semantische verwantschap tussen woorden en over het algemeen scoorde de cosinus het best.

3.3 Toepassingen

In [20] en [27] staan heel wat toepassingen van LSA beschreven. Hier volgt een kort overzicht van de belangrijkste:

- **Tekstgebaseerde informatie-ontsluiting**: Een gebruiker geeft een query in met enkele kernwoorden en zou graag een lijst terugkrijgen met de documenten die het best bij deze kernwoorden horen. Hiervoor kan LSA gebruikt worden. De query wordt voorgesteld als een pseudodocument, dit is een gewogen gemiddelde van de vectoren van de woorden die het bevat. Vervolgens wordt de cosinus tussen dit pseudodocument en alle documenten uit het corpus berekend en worden ze gerangschikt. De documenten met de hoogste cosinuswaarde worden teruggegeven aan de gebruiker. LSA gebruikt in deze context wordt ook wel Latent Semantisch Indexeren (LSI) genoemd.

- **Betekenisverwantschap tussen woorden**: LSA slaagt erin om verborgen betekenisverwantschap tussen woorden te vinden. Dit kan nuttig zijn om synoniemen van een woord te ontdekken. Aangezien synoniemen dikwijls niet in dezelfde documenten voorkomen, is het niet zo eenvoudig om ze uit grote

Woord	**cos(*Kuifje*,woord)**	**Woord**	**cos(*Kuifje*,woord)**
Hergé	0.9579	Tintin	0.9151
Haddock	0.8875	Kuifjes	0.8833
Hergés	0.8799	tekenaar	0.8310
stripverhaal	0.8161	strip	0.8119
Moulinsart	0.8118	Hergé's	0.8104
strips	0.8053	Molensloot	0.8028
stripgeschiedenis	0.7950	striptekenaar	0.7868
Robbedoes	0.7860	striptekenaars	0.7815
Franquin	0.7805	beeldverhaal	0.7754
Mortimer	0.7741	Guust	0.7740

Tabel 3.1: De twintig woorden die het dichtst bij *Kuifje* liggen in een LSA-model met 100 dimensies, getraind op data uit De Standaard. Als metriek is de cosinus tussen twee woorden gebruikt.

hoeveelheid tekst te vinden. LSA is reeds gebruikt om het synoniemgedeelte van de Engelse taaltest TOEFL op te lossen. In het synoniemgedeelte van deze test is steeds een woord gegeven en moet de ondervraagde een synoniem voor dit woord geven of een woord dat er het meeste op lijkt qua betekenis. Hij kan hierbij steeds kiezen uit een beperkte lijst van mogelijke antwoorden. Om dit probleem op te lossen met LSA volstaat het om de cosinus tussen het gegeven woord en alle antwoordmogelijkheden te berekenen. Als antwoord wordt dan het woord behorend bij de grootste cosinus gegeven.

Naast synoniemen kan LSA ook alle woorden terugvinden die semantisch verwant zijn met een bepaald woord. Als voorbeeld hiervan is een LSA-model met dimensie $k = 100$, getraind op data uit De Standaard, gebruikt om de woorden terug te vinden die het dichtst bij het woord *Kuifje* liggen in de gereduceerde ruimte. De cosinus tussen twee woorden is hierbij gebruikt als metriek en de twintig woorden met de hoogste cosinuswaardes zijn gezocht. Deze woorden zijn samen met de cosinuswaardes terug te vinden in tabel 3.1. Naast logische woorden als *Hergé*, *Haddock* of *Tintin* zijn er ook enkele interessante woorden bij, zoals *Robbedoes*, *Franquin* of *Mortimer*. Deze woorden duiden op andere stripreeksen dan *Kuifje* en zullen dus niet altijd in dezelfde documenten voorkomen. Ze komen echter wel in een gelijkaardige context voor, namelijk stripverhalen. LSA heeft als het ware de verborgen semantische verwantschap tussen deze woorden ontdekt.

- **Document clustering**: Het clusteren van documenten of woorden kan uitgevoerd worden met LSA. De cosinus tussen twee vectoren is een maat voor het onderling verwantschap. Binnen een zelfde cluster zal het verwantschap hoog liggen, terwijl tussen de clusters onderling het verwantschap klein is. De

standaard clusteringalgoritmen kunnen hier gebruikt worden.

- **Document classificatie**: Een classificatieprobleem is een probleem waar een set van documenten met gekend label gegeven is en een set van documenten met ongekend label. De opdracht is nu om uit de trainingsset te leren om een label toe te kennen aan de documenten met ongekend label. Een voorbeeld van zo'n label kan het onderwerp van het document zijn. LSA is zeer geschikt voor deze opdracht. Een mogelijke oplossing bestaat erin de cosinus tussen alle documenten uit de trainingsset en het nieuwe document te berekenen. Het nieuwe document zal dan hetzelfde label toegewezen krijgen als het document waarvan de cosinus met het nieuwe document het grootste was.

- **Toekennen van essayscores**: Een andere toepassing van LSA is het automatisch toekennen van scores op essays. Dit kan gebruikt worden wanneer studenten een essay moeten schrijven over een bepaald thema. De leraar voorziet een voorbeeldessay waarin de belangrijkste elementen staan. De essays van de studenten worden dan vergeleken met het voorbeeldessay. De cosinus tussen de twee essays is een maat voor de kwaliteit van het essay. De essays met een hoge cosinuswaarde krijgen de hoogste scores.

3.4 LSA-taalmodel

Zoals gezien in de vorige sectie, biedt LSA heel wat toepassingen. In deze masterproef is echter enkel het gebruik van LSA binnen taalmodellering van belang. In deze sectie komt de uitwerking hiervan aan bod.

3.4.1 Keuze van de context

Net als bij N-gram taalmodellen is het doel een uitwerking te vinden voor $P(w_q|H_{q-1})$. Hierbij is w_q het te voorspellen woord en is H_{q-1} de toegelaten geschiedenis of context. Bij N-gram taalmodellen is $P(w_q|H_{q-1}) = P(w_q|w_{q-1}, w_{q-2}, ..., w_{q-N+1})$. De kansberekening is bij het LSA-model:

$$P(w_q|H_{q-1}) = P(w_q|H_{q-1}, \mathcal{S}) \tag{3.21}$$

De afhankelijkheid van $\mathcal{S}$ verwijst naar het feit dat de kansberekening zal afhangen van de vectorruimte $\mathcal{S}$ met bijbehorende dimensie k.
Het volgende dat nu moet gebeuren is het bepalen van de context H_{q-1}. In elk geval moet de context causaal zijn en dus afgebroken worden na het woord w_{q-1}. Voor de rest is de vrijheid echter groot bij LSA-modellering. Nergens eerder werd er een beperking opgelegd aan de context die mag gebruikt worden. De context mag zowel het vorige woord zijn, de $N-1$ vorige woorden, de huidige zin, het huidige document of zelfs het hele trainingscorpus. Deze ruime keuze is een groot voordeel van LSA-taalmodellering. Meestal wordt als context de vorige woorden in hetzelfde document genomen. Zonder verlies aan algemeenheid wordt dit verder verondersteld.

Nu de context H_{q-1} bepaald is, rest nog een methode te specifiëren om 3.21 te berekenen. Dit vormt het onderwerp van de rest van deze sectie.

3.4.2 Pseudodocumentvoorstelling

Uit de afgebroken SWO (3.7) van de frequentiematrix W volgt:

$$w_i \approx u_i S V^T \tag{3.22}$$

De subindices k die wijzen op de afgebroken SWO zijn omwille van de leesbaarheid weggelaten. Zonder verlies aan algemeenheid kan het benaderingssymbool vervangen worden door een gelijkheidsteken. De linker singuliere vector u_i kan hieruit gehaald worden:

$$u_i = w_i V S^{-1} \tag{3.23}$$

Op analoge wijze kan een uitdrukking gevonden worden voor de rechter singuliere vector v_j. Het document d_j kan geschreven worden als:

$$d_j = U S v_j^T \tag{3.24}$$

De rechter singuliere vector v_j is dan gelijk aan:

$$v_j = d_j^T U S^{-1} \tag{3.25}$$

Stel dat er nu een nieuw document $\tilde{d}_p^T$ is dat niet gebruikt werd bij de constructie van $\mathcal{S}$. Nu is er een manier nodig om dit document in de vectorruimte $\mathcal{S}$ voor te stellen. Dit kan met behulp van formule 3.25:

$$\tilde{v}_p = \tilde{d}_p^T U S^{-1} \tag{3.26}$$

De vector $\tilde{v}_p$ is nu de voorstelling van het nieuwe document $\tilde{d}_p^T$ in de ruimte $\mathcal{S}$. Omdat het nieuwe document $\tilde{d}_p^T$ geen deel uitmaakt van W, wordt het een pseudodocument genoemd [3]. De bijbehorende voorstelling $\tilde{v}_p$ in de ruimte $\mathcal{S}$ is de pseudodocumentvoorstelling.

3.4.3 Directe modellering

Om $P(w_q|H_{q-1}, \mathcal{S})$ te modelleren moet de context H_{q-1} kunnen voorgesteld worden in de ruimte $\mathcal{S}$. Dit kan door de context als een pseudodocument te beschouwen: $H_{q-1} = \tilde{d}_{q-1}$. Nu kan vergelijking 3.26 gebruikt worden om de pseudodocumentvoorstelling $\tilde{v}_{q-1} \in \mathcal{S}$ te vinden. Het taalmodel is nu:

$$P(w_q|H_{q-1}, \mathcal{S}) = P(w_q|\tilde{d}_{q-1}) \tag{3.27}$$

Dit kan berekend worden uit de voorstellingen van w_q en $\tilde{d}_{q-1}$ in $\mathcal{S}$. Met andere woorden, de kans op optreden van w_q, gegeven $\tilde{d}_{q-1}$, kan afgeleid worden uit de afstand tussen u_q en $\tilde{v}_{q-1}$ in $\mathcal{S}$.

De afstand tussen een woord en een document kan op een analoge manier afgeleid worden als de afstand tussen woorden of documenten onderling. Het (i, j)-de element van W is een maat voor in hoe verre het woord w_i en het document d_j samen voorkomen. Uit de SWO van W volgt dat het (i, j)-de element van W gelijk is aan het inwendig product tussen $u_i S^{1/2}$ en $v_j S^{1/2}$. Dit is gelijk aan de cosinus van de hoek tussen de vectoren $u_i S^{1/2}$ en $v_j S^{1/2}$:

$$K(u_q, \tilde{v}_{q-1}) = \cos(u_q S^{1/2}, \tilde{v}_{q-1} S^{1/2}) \tag{3.28}$$

$$= \frac{u_q S \tilde{v}_{q-1}^T}{\|u_q S^{1/2}\| \, \|\tilde{v}_{q-1} S^{1/2}\|} \tag{3.29}$$

Een waarde van $K(u_q, \tilde{v}_{q-1})$ dicht bij 1 betekent dat $\tilde{d}_{q-1}$ een sterke voorspeller is van w_q. Daartegenover staat dat een lage waarde van $K(u_q, \tilde{v}_{q-1})$ wil zeggen dat $\tilde{d}_{q-1}$ geen goede voorspeller is van w_q.

Nu er al een manier is gevonden om de mate van verwantschap tussen een woord en een document te vinden, is de definitieve kansschatting niet meer zo ver. Een mogelijkheid is om de cosinus tussen het volgende woord w_q en het pseudodocument $\tilde{d}_{q-1}$ in de ruimte $\mathcal{S}$ te berekenen. Normaliseren gebeurt door te delen door de som van de cosinussen van alle woorden uit het vocabularium $\mathcal{V}$ en het pseudodocument $\tilde{d}_{q-1}$. Het resultaat is echter nog geen kans, want de cosinus kan negatief zijn. De oplossing bestaat erin van elke cosinuswaarde de minimum cosinuswaarde af te trekken, dit is de kleinste cosinuswaarde die gevonden is van alle woorden in het vocabularium $\mathcal{V}$ met het pseudodocument $\tilde{d}_{q-1}$:

$$P(w_q|\tilde{d}_{q-1}) = \frac{K(u_q, \tilde{v}_{q-1}) - K_{\min}(u, \tilde{v}_{q-1})}{\sum_{w_i \in \mathcal{V}} K(u_i, \tilde{v}_{q-1}) - K_{\min}(u, \tilde{v}_{q-1})} \tag{3.30}$$

Het dynamisch bereik van deze LSA-kansen is echter zeer laag. De afgeleide kansen zullen dicht rond $1/|\mathcal{V}|$ schommelen, waardoor het LSA-model een matige voorspeller zal zijn. Een groter bereik ontstaat door het cosinusverschil tot een exponent γ te verheffen, met $\gamma > 1$ en typisch een geheel getal [8] [23]:

$$P(w_q|\tilde{d}_{q-1}) = \frac{[K(u_q, \tilde{v}_{q-1}) - K_{\min}(u, \tilde{v}_{q-1})]^\gamma}{\sum_{w_i \in \mathcal{V}} [K(u_i, \tilde{v}_{q-1}) - K_{\min}(u, \tilde{v}_{q-1})]^\gamma} \tag{3.31}$$

Deze kans zal het hoogst zijn voor de woorden waarvan de betekenis het dichtste ligt bij de context. Bijgevolg zal de kans hoog zijn voor inhoudswoorden, die meestal een eigen betekenis hebben, en het laagst voor functiewoorden, die weinig vertellen over de inhoud van een document. Aangezien er veel meer functiewoorden zijn dan inhoudswoorden zal dit voor het LSA-taalmodel leiden tot een hoge waarde van de perplexiteit. Dit wil niet zeggen dat het LSA-model niet te gebruiken is in taalmodellering. In combinatie met andere modellen kan het een sterke aanvulling zijn. De meest voorkomende combinatie is die met N-grammen. Deze combinatie vormt het onderwerp van sectie 3.6.

3.5 Sterktes en zwaktes van LSA

Het LSA-taalmodel heeft zowel sterke als zwakke punten. Deze zullen hier aan bod komen. Als referentiemodel wordt het N-gram taalmodel genomen.
De voornaamste sterke punten van LSA zijn de volgende:

- In tegenstelling tot N-grammen is het afhankelijkheidssgebied bij LSA-taalmodellen heel groot. In principe kan de context willekeurig gekozen worden, maar meestal gaat de voorkeur naar de woorden in hetzelfde document voorafgaand aan het huidige woord. Door de hele documentgeschiedenis in rekening te brengen, kunnen alle belangrijke woorden, die eerder in het document verschenen zijn, mee gebruikt worden om het volgende woord te voorspellen.

- N-gram taalmodellen houden enkel rekening met de woordvolgorde bij voorspelling van het volgende woord. Ze nemen de betekenis van de vorige woorden niet in achting. Dit doet het LSA-taalmodel wel. Deze voorspelt het volgende woord op basis van de betekenis van de vorige woorden. Het geeft een schatting van de kans op het volgende woord terug uit de afstand van het huidige woord tot de documentgeschiedenis in de gereduceerde ruimte.

- De gereduceerde ruimte waarin LSA werkt is een betere voorstelling van documenten en woorden. Zoals eerder vermeld in sectie 3.2.3 zorgt een dimensionaliteitsreductie voor vermindering van ruis in de frequentiematrix W. Nog een voordeel van het gebruik van deze gereduceerde ruimte is dat zowel woorden als documenten zich nu in dezelfde ruimte bevinden en zo gemakkelijk met elkaar te vergelijken zijn.

- Ten slotte nog als laatste voordeel is dat LSA heel goed synoniemen kan opvangen. In de gereduceerde ruimte liggen synoniemen dicht bij elkaar en zo kan LSA dit opmerken. Daar waar voor N-grammen de zinnen *De kat liep buiten* en *De poes liep buiten* compleet verschillend zijn, zullen deze zinnen door het LSA-model zo goed als gelijkwaardig beschouwd worden.

Het LSA-taalmodel kent echter ook enkele zwakke punten:

- LSA-taalmodellen houden geen rekening met de woordvolgorde, waardoor syntactisch foute constructies kunnen ontstaan. Het LSA-taalmodel zal bv. wegens de grote overeenkomst tussen *hond* en *kat* een grote kans toewijzen aan het woord *kat* na *De hond liep naar*, terwijl dit syntactisch fout is. Zulke lokale afhankelijkheden worden wel goed gemodelleerd door N-gram taalmodellen. Dit wijst erop dat een combinatie van N-gram taalmodellen, die lokale afhankelijkheden modelleren, met het LSA-taalmodel, dat globale afhankelijkheden modelleert, mogelijk een beter model zou kunnen opleveren.

- Een ander zwak punt van LSA-taalmodellen is de rekencomplexiteit. Het LSA-model moet eerst een SWO berekenen, wat computationeel intensief is. Eens het model berekend is, moet het om een woord te voorspellen de

LSA-kans berekenen voor alle woorden uit het vocabularium. Dit vraagt heel wat meer werk dan bij N-gram taalmodellen, waar eens het model gekend is, enkele opzoekingen in een tabel volstaan om de kans op het volgende woord te schatten.

- Bij het afbreken van de SWO minimaliseert de resulterende matrix $\hat{W}$ de uitdrukking $\left\|\hat{W} - W\right\|_F$ over alle matrices $\hat{W}$ van rang k. Het minimaliseren van de Frobeniusnorm zal de ruis minimaliseren op voorwaarde dat deze een Gaussische verdeling heeft. Het staat echter vast dat woordfrequenties geen Gaussische verdeling hebben [27].

- Een laatste zwak punt van LSA-taalmodellen heeft te maken met polysemie. Polysemie geeft aan dat eenzelfde woord verschillende betekenissen kan hebben. In het LSA-taalmodel komt elk woord uit het vocabularium $\mathcal{V}$ maar één keer voor, waardoor de verschillende voorkomens van dit woord in de documenten, al dan niet in een andere betekenis, als hetzelfde worden beschouwd. Bijgevolg kan het LSA-model geen polysemie opvangen. Zo is er voor het LSA-taalmodel bv. geen onderscheid tussen een bank om op te zitten en een bank als financiële instelling.

3.6 Combinatie met N-grammen

Het LSA-taalmodel houdt op geen enkele manier rekening met de woordvolgorde, in tegenstelling tot N-grammen. Hierdoor slaagt het er niet in om lokale afhankelijkheden te modelleren en is het in termen van perplexiteit een slecht taalmodel. Het LSA-taalmodel modelleert echter wel globale afhankelijkheden, en dit is een eigenschap die N-grammen niet bezitten. Het is echter wenselijk om zowel de lokale afhankelijkheden, voorzien door N-grammen, te combineren met de globale afhankelijkheden, gemodelleerd door het LSA-taalmodel. Hiervoor zijn verschillende modellen mogelijk. In de eerste subsectie wordt een model afgeleid vanuit een Bayesiaanse interpretatie. In de daaropvolgende subsectie komen andere modellen aan bod. Hierbij zitten voornamelijk modellen die zullen werken met een LSA betrouwbaarheidsfactor. Deze factor geeft aan hoe betrouwbaar de LSA-schatting is. Subsectie 3.6.1 is grotendeels gebaseerd op [3].

3.6.1 Bayesiaanse interpretatie

Het doel is een uitdrukking te vinden om het volgende te berekenen:

$$P(w_q|H_{q-1}) = P(w_q|H_{q-1}^{(n)}, H_{q-1}^{(l)}) \tag{3.32}$$

De context H_{q-1} bestaat nu uit een N-gram component $\left[H_{q-1}^{(n)} = w_{q-1}w_{q-2}...w_{q-n+1}\right]$ en een LSA component $\left[H_{q-1}^{(l)} = \tilde{d}_{q-1}\right]$. Door marginale expansie toe te passen over

alle woorden w_i uit het vocabularium $\mathcal{V}$ kan vergelijking 3.32 geschreven worden als:

$$P(w_q|H_{q-1}) = \frac{P(w_q, H_{q-1}^{(l)}|H_{q-1}^{(n)})}{\sum_{w_i \in \mathcal{V}} P(w_i, H_{q-1}^{(l)}|H_{q-1}^{(n)})} \tag{3.33}$$

De teller kan verder ontwikkeld worden:

$$P(w_q, H_{q-1}^{(l)}|H_{q-1}^{(n)}) = P(w_q|H_{q-1}^{(n)}) \cdot P(H_{q-1}^{(l)}|w_q, H_{q-1}^{(n)}) \tag{3.34}$$

$$= P(w_q|w_{q-1}w_{q-2}...w_{q-n+1}) \cdot P(\tilde{d}_{q-1}|w_q w_{q-1}w_{q-2}...w_{q-n+1}) \tag{3.35}$$

Nu wordt de veronderstelling gemaakt dat de kans van de documentgeschiedenis gegeven het huidige woord onafhankelijk is van de onmiddellijke context voorafgaand aan dit woord. Voor een woord kunnen er immers verschillende syntactische constructies (de onmiddellijke context voorafgaand) gebruikt worden om dezelfde betekenis te dragen. De kansberekening wordt dan:

$$P(w_q|H_{q-1}) = \frac{P(w_q|w_{q-1}w_{q-2}...w_{q-n+1})P(\tilde{d}_{q-1}|w_q)}{\sum_{w_i \in \mathcal{V}} P(w_i|w_{q-1}w_{q-2}...w_{q-n+1})P(\tilde{d}_{q-1}|w_i)} \tag{3.36}$$

Deze vergelijking heeft een quasi-Bayesiaanse interpretatie. $P(\tilde{d}_{q-1}|w_q)$ kan gezien worden als een prior. Vergelijking 3.36 is dan de klassieke Bayesiaanse schatting van de N-gram kans, gebruikt met een prior verkregen uit het LSA-model.
De uitdrukking $P(\tilde{d}_{q-1}|w_q)$ kan met de regel van Bayes herschreven worden als:

$$P(\tilde{d}_{q-1}|w_q) = \frac{P(w_q|\tilde{d}_{q-1})P(\tilde{d}_{q-1})}{P(w_q)} \tag{3.37}$$

Vergelijking 3.36 wordt dan:

$$P(w_q|H_{q-1}) = \frac{P(w_q|w_{q-1}w_{q-2}...w_{q-n+1})\frac{P(w_q|\tilde{d}_{q-1})P(\tilde{d}_{q-1})}{P(w_q)}}{\sum_{w_i \in \mathcal{V}} P(w_i|w_{q-1}w_{q-2}...w_{q-n+1})\frac{P(w_i|\tilde{d}_{q-1})P(\tilde{d}_{q-1})}{P(w_i)}} \tag{3.38}$$

$$= \frac{P(w_q|w_{q-1}w_{q-2}...w_{q-n+1})P(w_q|\tilde{d}_{q-1})}{P(w_q)\sum_{w_i \in \mathcal{V}} \frac{P(w_i|w_{q-1}w_{q-2}...w_{q-n+1})P(w_i|\tilde{d}_{q-1})}{P(w_i)}} \tag{3.39}$$

$$= \frac{P_{N\text{-gram}}(w_q|w_{q-1}...w_{q-n+1})P\,(w_q|\tilde{d}_{q-1})}{P_{\text{uni}}(w_q)\sum_{w_i \in \mathcal{V}} \frac{P_{N\text{-gram}}(w_i|w_{q-1}...w_{q-n+1})P_{\text{LSA}}(w_i|\tilde{d}_{q-1})}{P_{\text{uni}}(w_i)}} \tag{3.40}$$

aangezien $P(\tilde{d}_{q-1})$ onafhankelijk is van w_i kan het geschrapt worden in teller en noemer. De term $P_{\text{uni}}(w_i)$ is de unigram kans van woord w_i. Alle termen in teller en noemer kunnen berekend worden en dat maakt deze uitdrukking bruikbaar als taalmodel.

3.6.2 Andere modellen

Het model afgeleid in de vorige sectie is echter niet de enige mogelijkheid om LSA te combineren met N-gram taalmodellen. De eenvoudigste methode is simpelweg een lineaire interpolatie (LI) tussen de N-gram kans en de LSA-kans:

$$P_{\text{LI}}(w_q|H_{q-1}) = \lambda P_{\text{LSA}}(w_q|\tilde{d}_{q-1}) + (1-\lambda)P_{N\text{-gram}}(w_q|w_{q-1}...w_{q-n+1}) \tag{3.41}$$

Er geldt steeds dat $0 \leq \lambda \leq 1$. De interpolatieparameter λ wordt bepaald door het minimaliseren van de perplexiteit op een validatieset.

Deze aanpak is echter te simplistisch, want bij bepaalde woorden zal de LSA-schatting heel slecht zijn en bij andere woorden zal de N-gram schatting slecht zijn. Het zou beter zijn om deze interpolatieparameter te laten variëren afhankelijk van de betrouwbaarheid van de LSA-kans. Zoals eerder gezegd zijn LSA-modellen goede schatters van inhoudswoorden en slechte schatters van functiewoorden. Beschouw hierom een betrouwbaarheidsfactor λ_i van het woord w_i. Deze factor geeft aan in hoe verre de LSA-kans betrouwbaar is of niet. De factor wordt gedefinieerd als:

$$\lambda_i = \frac{1}{2}\left[1 + \frac{1}{\log(N)}\sum_{j=1}^{N} f_{ij}\log f_{ij}\right] \tag{3.42}$$

met N het aantal documenten in het trainingscorpus en $f_{ij} = c_{ij}/t_i$, met c_{ij} het aantal keer dat woord i in document j voorkomt en t_i het aantal keer dat woord i in heel het corpus voorkomt. De term binnen de haakjes van vergelijking 3.42 is de negatieve genormaliseerde entropie E_i, zie sectie 3.2.2, van woord w_i en dit opgeteld met 1:

$$\lambda_i = \frac{1}{2}\left[1 - E_i\right] \tag{3.43}$$

De betrouwbaarheidsfactor zal hoog zijn voor inhoudswoorden (lage E_i) en laag voor functiewoorden (hoge E_i). De factor $\frac{1}{2}$ voor de haakjes zorgt ervoor dat de betrouwbaarheidsfactor λ_i niet groter kan worden dan 0.5. Een nieuw model kan nu afgeleid worden waarin de betrouwbaarheidsfactor λ_i als interpolatieparameter dient:

$$P_{\text{IWAM}}(w_q|H_{q-1}) = \frac{\lambda_q P_{\text{LSA}}(w_q|\tilde{d}_{q-1}) + (1-\lambda_q)P_{N\text{-gram}}(w_q|w_{q-1}...w_{q-n+1})}{\sum_{w_i\in\mathcal{V}} \lambda_i P_{\text{LSA}}(w_i|\tilde{d}_{q-1}) + (1-\lambda_i)P_{N\text{-gram}}(w_i|w_{q-1}...w_{q-n+1})} \tag{3.44}$$

Dit model wordt *information weighted arithmetic mean* (IWAM) genoemd [8]. Let hierbij ook op dat de LSA-schatting nooit meer dan een gewicht van 0.5 zal krijgen wegens de bovenlimiet van de betrouwbaarheidsfactor λ_q.

In [8] staat beschreven dat een niet-lineaire combinatie tussen de LSA- en N-gram schatting leidt tot betere resultaten. Bij een lineaire combinatie tussen de twee kan het gebeuren dat de LSA-schatter woorden voorspelt die syntactisch fout zijn. Door de optelling speelt de lage N-gram schatting geen rol en domineert de LSA-kans. Stel bv. dat voor een woord geldt dat $P_{\text{LSA}} = 10^{-3}$, $P_{N\text{-gram}} = 10^{-9}$ en $\lambda_q = 0.1$. Dan

is $0.1P_{\text{LSA}} + 0.9P_{N\text{-gram}} \approx 10^{-4}$. De N-gramschatting heeft hier nauwelijks invloed, hoewel deze waarschijnlijk aangeeft dat het woord syntactisch niet correct is. Als nu echter voor een woord $P_{N\text{-gram}} = 10^{-4}$, dan zou dit woord syntactisch wel correct kunnen zijn. De gecombineerde kans is nu $\approx 2.10^{-4}$, nauwelijks een factor 2 groter dan het voorbeeld met een lage N-gram schatting.
Ook gebeurt het dat als beide modellen een goede schatting geven, dit door de optelling te weinig tot uiting komt in de definitieve schatting. Een oplossing is om beide schattingen eerst tot de betrouwbaarheidsfactor te verheffen en vervolgens te vermenigvuldigen met elkaar:

$$P_{\text{IWGM}}(w_q|H_{q-1}) = \frac{P_{\text{LSA}}^{\lambda_q}(w_q|\tilde{d}_{q-1})P_{N\text{-gram}}^{(1-\lambda_q)}(w_q|w_{q-1}...w_{q-n+1})}{\sum_{w_i\in\mathcal{V}} P_{\text{LSA}}^{\lambda_i}(w_i|\tilde{d}_{q-1})P_{N\text{-gram}}^{(1-\lambda_i)}(w_i|w_{q-1}...w_{q-n+1})} \tag{3.45}$$

Dit model staat bekend als *information weighted geometric mean* (IWGM) [8].

Nog een andere niet-lineaire combinatie staat bekend als *similarity modulated N-gram* (SIMMOD) [23]:

$$P_{\text{SIMMOD}}(w_q|H_{q-1}) = \frac{P_{\text{LSA}}(w_q|\tilde{d}_{q-1})P_{N\text{-gram}}(w_q|w_{q-1}...w_{q-n+1})}{\sum_{w_i\in\mathcal{V}} P_{\text{LSA}}(w_i|\tilde{d}_{q-1})P_{N\text{-gram}}(w_i|w_{q-1}...w_{q-n+1})} \tag{3.46}$$

Deze laatste vorm komt overeen met de Bayesiaanse interpretatie waarbij $P(w)$ uniform verondersteld is.
Dat een niet-lineaire combinatie tussen N-grammen en LSA het waarschijnlijk beter doet, kan aangetoond worden met het voorbeeld van hierboven. Als voor een woord geldt dat $P_{\text{LSA}} = 10^{-3}$ en $P_{N\text{-gram}} = 10^{-9}$, dan is de gecombineerde kans 10^{-12} bij SIMMOD. Wanneer echter voor een ander woord $P_{N\text{-gram}} = 10^{-4}$, leidt dit bij SIMMOD tot een gecombineerde kans van 10^{-7}. Dit is een factor 10^5 hoger dan voor het eerste woord. In dit model zal het syntactisch foute eerste woord een verwaarloosbare kans krijgen tegenover het syntactisch correcte tweede woord.

3.7 N-grammen in LSA

In de vorige sectie werden verschillende combinatiemodellen tussen N-grammen en LSA besproken. Zowel het N-gram als het LSA-model werden echter onafhankelijk van elkaar aangemaakt. Het idee bestaat om bij het aanmaken van een taalmodel beide te combineren. Een mogelijkheid is om in de kolommen van de frequentiematrix W, gebruikt bij LSA, de documenten te vervangen door een korte context. Als context kan bijvoorbeeld gekozen worden voor de twee voorgaande woorden, de twee volgende woorden of het vorige en het volgende woord. Uitbreidingen naar grotere contexten zijn ook mogelijk.
Neem nu dat de context de twee vorige woorden voorstelt. Dat wilt zeggen dat de twee eerste woorden (context d_j) van alle 3-grammen uit de trainingsset in de kolommen van de frequentiematrix zullen gestoken worden. In de rijen staan nog steeds alle woorden w_i uit het vocabularium $\mathcal{V}$. Dan is het (i, j)-de element van de

Woord
Fransen
zuiderburen
Frankrijks
Franse
Italië
Spanje
niet-Franse
Zwitserland
Vichy
Français
Parijs
Mer
Vendée
Leclerc
Noord-Franse
Bretagne
Noord-Frankrijk
Glavany
Thionville
Martinique

(a) Klassieke aanpak

Woord
Groot-Brittannië
Nederland
Duitsland
Zwitserland
Australië
België
Wallonië
Spanje
Oostenrijk
Brazilië
Vlaanderen
Italië
Marokko
West-Vlaanderen
Noorwegen
Saudi-Arabië
Madrid
Denemarken
Afghanistan
Hongarije

(b) N-grammen in LSA

Woord
Italië
Zwitserland
Spanje
Oostenrijk
Duitsland
Denemarken
Ierland
Finland
Noorwegen
Luxemburg
Hongarije
Griekenland
Nederland
Zweden
Groot-Brittannië
België
Portugal
Roemenië
Beieren
Bern

(c) Combinatie

Tabel 3.2: De twintig woorden die het dichtst bij *Frankrijk* liggen, volgens a) de klassieke aanpak, b) N-grammen in LSA en c) combinatie van beide.

frequentiematrix gelijk aan het aantal keer dat woord w_i voorkomt na de context d_j. Aangezien het aantal contexten van de orde $|\mathcal{V}|^2$ is, zal dit leiden tot een zeer grote en spaarse matrix. Dan volgt dezelfde procedure als bij gewone LSA: het wegen van de elementen, dit gebeurt op dezelfde manier als voorheen, en vervolgens het berekenen van de afgebroken SWO. Nadien is het terug mogelijk om een cosinus tussen woorden en/of contexten te berekenen. Als deze woorden in gelijkaardige contexten voorkomen zal de cosinus tussen twee woorden hoog liggen. Hier zullen dus niet altijd woorden die semantisch gelijkaardig zijn in dezelfde richting liggen, maar ook woorden die syntactisch gelijkaardig zijn.

Als voorbeeld is in tabel 3.2 het onderscheid tussen beide aanpakken geïllustreerd. De twintig woorden, die het dichtst bij *Frankrijk* in de gereduceerde ruimte liggen, zijn weergegeven in afnemende cosinus voor beide modellen. In de linkse kolom is het resultaat te zien van het klassieke LSA-model. Naast andere landen en Franse streken zijn er ook woorden zoals *Fransen*, *Frankrijks* en *Noord-Franse* die dicht bij het woord *Frankrijk* liggen. De middelste kolom toont het resultaat van de N-grammen ingewerkt in de rijen en kolommen van LSA. Hier valt op dat alle woorden die dicht-

Woord
kruidenier
kapper
grenscontrole
notaris
apotheker
dealer
slager
onthaalmoeder
kleermaker
gynaecoloog
winkelaar
huisarts
onderzoeksrechter
belastingcontroleur
juwelier
verkoper
beenhouwer
beursgang
exploitant
N-VA'er

Tabel 3.3: De twintig woorden die het dichtst bij *bakker* liggen volgens de methode van N-grammen in LSA.

bij *Frankrijk* liggen, syntactisch gelijk zijn. Het zijn eveneens allemaal eigennamen, sterker nog het zijn allemaal landen, steden of streken. Dit resultaat valt zeer goed mee, aangezien het gewenst is om naast syntactisch gelijke ook nagenoeg semantisch gelijke woorden in de lijst te krijgen.
De resultaten zijn echter niet voor alle woorden even goed. Vooral voor eigennamen werkt deze methode goed. Een voorbeeld van een minder goed resultaat is te zien in tabel 3.3. Daar zijn de twintig woorden gegeven het dichtste bij *bakker*. De meeste andere woorden zijn eveneens beroepen, wat gewenst is. Er staan echter ook enkele woorden in de lijst die weinig of niets met *bakker* te maken hebben, zoals *grenscontrole* of *beursgang*.

Een ander idee is om beide aanpakken te combineren binnen eenzelfde model. Dit kan door zowel de documenten als de korte contexten in de kolommen van de frequentiematrix onder te brengen. Hierbij dient wel een groter gewicht toegekend te worden aan de documenttellingen, aangezien er veel minder documenten zijn dan korte contexten. In de rechtse kolom van tabel 3.2 zijn nogmaals de twintig woorden gegeven die het dichtst bij Frankrijk liggen, nu voor het gecombineerde model. Het resultaat is dat de eerste 18 plaatsen door landen worden ingenomen, dit zijn woorden die zowel syntactisch gelijk als semantisch zeer verwant zijn. Het valt op dat de woorden

Italië, *Zwitserland* en *Spanje* bovenaan staan, deze woorden scoorden zowel hoog bij het klassieke LSA-model als bij het model met de korte contexten in de kolommen.

Hoe kan dit resultaat gebruikt worden voor de aanmaak van een taalmodel? Stel nu dat er in de trainingsset veel voorbeelden voorkomen van het 3-gram *ging naar Zwitserland* en dat het 3-gram *ging naar Spanje* nauwelijks in de trainingsset voorkomt. Dan zal $P(Zwitserland|ging, naar)$ groot zijn in het 3-gram taalmodel en $P(Spanje|ging, naar)$ klein. Deze twee 3-grammen zouden echter een ongeveer even grote kans moeten toegekend krijgen. Een mogelijke oplossing is dat het 3-gram *ging naar Zwitserland* een deel van zijn kansmassa afstaat aan *ging naar Spanje*. Dit mag enkel indien *Zwitserland* en *Spanje* dicht bij elkaar liggen in de gereduceerde ruimte van het nieuwe LSA-model. Deze methode kan het spaarsheidprobleem van N-grammen oplossen. Een goede strategie dient echter ontwikkeld te worden voor het afstaan van een deel van de kansmassa door een woord aan andere dichtbijzijnde woorden. Dit valt buiten het bestek van deze masterproef, maar kan wel een inspiratiebron zijn voor verder onderzoek.

Hoofdstuk 4

Probabilistische latent semantische analyse

4.1 Inleiding

Een groot minpunt van LSA-taalmodellen is dat het niet met polysemie kan omgaan. Ook gaat LSA door het minimaliseren van een Frobeniusnorm ervan uit dat woordtellingen Gaussisch verdeeld zijn. Deze twee problemen komen niet voor bij probabilistische latent semantische analyse (pLSA). In dit hoofdstuk komt deze techniek aan bod.

pLSA heeft veel overeenkomsten met LSA. Allereerst maakt het ook gebruik van een frequentiematrix om het taalmodel op te stellen. Nadien zal het een dimensionaliteitsreductie uitvoeren via een matrixfactorisatie.
In tegenstelling tot LSA bekijkt pLSA alles vanuit een statistisch oogpunt. Het beschrijft een klasse van statistische modellen waarin de semantische eigenschappen van woorden en documenten uitgedrukt zijn in termen van probabilistische topics. Het topic model is gebaseerd op het idee dat documenten een mengeling zijn van topics, en dat elk topic een kansverdeling over woorden is. Een topic model is een generatief model voor documenten, het specifieert een eenvoudige statistische procedure om documenten te genereren. Het voordeel van een statistisch model te gebruiken is dat standaardtechnieken uit statistiek kunnen toegepast worden voor modelfitting, modelselectie en complexiteitscontrole [16]. De parameters van het topic model worden berekend met behulp van een Expectation-Maximization (EM) algoritme. Op basis van het topic model kan dan een taalmodel worden afgeleid.

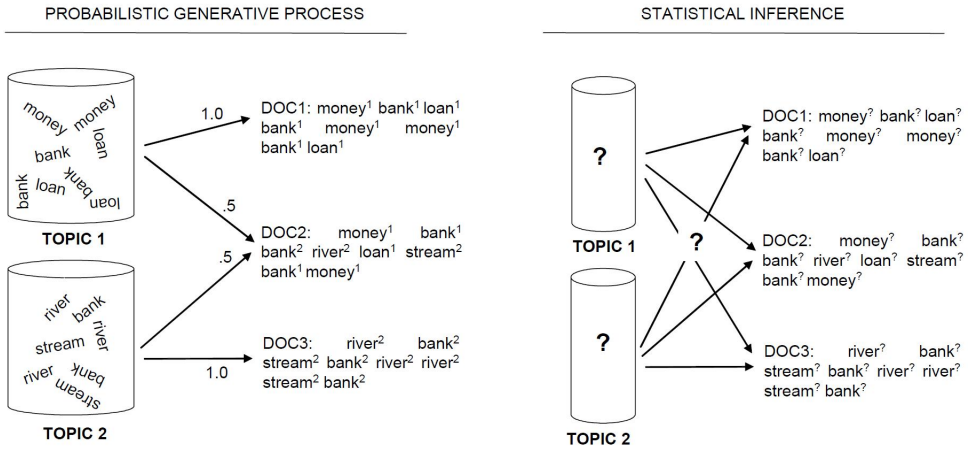
Het topic model komt aan bod in sectie 4.2. Vervolgens komt in sectie 4.3 het EM-algoritme aan bod dat het model opstelt. In sectie 4.4 wordt de uitwerking van pLSA tot een taalmodel gegeven samen met de combinatie met N-gram taalmodellen. Dan volgt in sectie 4.5 een discussie over pLSA. Om af te ronden gaat het in sectie 4.6 over pLSA als niet-negatieve matrixfactorisatie.

4.2 Topic model

4.2.1 Generatieve modellen

Een topic model is een generatief model voor documenten. Om een nieuw document te maken wordt er een uniforme verdeling over topics gekozen. Om dan een woord binnen het document te genereren, wordt er eerst een topic uit deze verdeling gekozen en vervolgens wordt binnen dat topic een woord gegenereerd. Bij het fitten van een generatief model is het doel om de beste selectie van topics of latente variabelen te vinden die de geobserveerde data beschrijven, ervan uitgaand dat het model de data gegenereerd heeft, dit is de maximum likelihood oplossing.

In de linkerhelft van figuur 4.1 wordt het topic model als generatief model geïllustreerd. Er zijn twee topics in totaal, het thema van het eerste is gerelateerd aan geldzaken en het thema van het tweede aan rivieren. Elk topic heeft een eigen verdeling van woorden. Documenten kunnen aangemaakt worden door woorden uit topics te selecteren volgens het gewicht toegekend aan het topic. Zo is bv. document 1 enkel samengesteld uit woorden afkomstig van topic 1, document 3 bestaat enkel uit woorden van topic 2 en document 2 is evenwichtig samengesteld uit woorden van zowel topic 1 als 2. Het superscript duidt aan uit welk topic het woord afkomstig is. Uit dit model blijkt dat hetzelfde woord door meerdere topics kan gegenereerd worden. Dit laat toe om het probleem van polysemie op te lossen. Een voorbeeld is *bank* dat zowel een financiële instelling als een rivieroever (in het Engels *river bank*) kan betekenen. Zowel topic 1 als 2 geven een hoge kans aan dit woord, maar het woord *bank* heeft binnen deze topics een andere betekenis.



Figuur 4.1: Illustratie van het generatieve proces en het probleem van statistische inferentie. Deze figuur komt uit [24].

De rechterhelft van figuur 4.1 illustreert het probleem van statistische inferentie. Gegeven zijn de geobserveerde woorden in een verzameling documenten. De vraag is nu welk topic model het meest waarschijnlijk is om deze data gegenereerd te hebben. Dit heeft te maken met het afleiden van volgende zaken: de kansverdeling

van woorden over topics en de kansverdeling van de topics over de documenten. Een formele beschrijving van dit model komt in de volgende subsectie aan bod.

4.2.2 Probabilistisch topic model

Topic modellen zijn gebaseerd op het idee dat documenten een mengeling zijn van topics, en dat elk topic een kansverdeling over woorden is. Stel dat er L topics zijn: $[t_1, t_2, ...t_L]$. De kans dat een woord geobserveerd wordt in document d_j is $P(d_j)$, $P(w_i|t_l)$ stelt een topicspecifieke kansverdeling over de woorden voor, en $P(t_l|d_j)$ stelt een documentspecifieke kansverdeling over de topics voor. Met behulp van deze definities kan het generatief model voor het samen voorkomen van woorden/documenten gedefinieerd worden door het volgende schema:

1. Selecteer een document d_j met kans $P(d_j)$
2. Neem een topic t_l met kans $P(t_l|d_j)$
3. Genereer een woord w_i met kans $P(w_i|t_l)$

Als resultaat wordt een observatiepaar (d_j, w_i) verkregen, terwijl het verkregen topic t_l weggelaten wordt. Het omzetten van dit generatief proces in een simultaan kansverdelingsmodel leidt tot de uitdrukking:

$$P(w_i, d_j) = P(d_j)P(w_i|d_j) \tag{4.1}$$

met

$$P(w_i|d_j) = \sum_{l=1}^{L} P(w_i|t_l)P(t_l|d_j) \tag{4.2}$$

Enkele voorbeelden van topics staan in tabel 4.1. Per topic zijn de twintig meest waarschijnlijke woorden gegeven samen met hun kans op optreden. In totaal zijn er 100 topics. Het valt op dat de topics meestal individueel interpreteerbaar zijn. Het thema van het eerste topic is boeken, het tweede thema handelt over de oorlog in Irak en in het derde topic is het thema media.

4.2.3 Interpretatie

Geometrische interpretatie: Het probabilistische topic model heeft een interessante geometrische interpretatie, die getoond wordt in figuur 4.2. Als het vocabularium $|\mathcal{V}|$ unieke woorden telt, dan is er een $|\mathcal{V}|$-dimensionale ruimte waarbij elke as de kans voorstelt van het observeren van een bepaald woord. De $(|\mathcal{V}| - 1)$-dimensionale simplex stelt alle kansverdelingen over de woorden voor. In figuur 4.2 is dit geïllustreerd voor drie woorden. Hierin is het grijze gebied de tweedimensionale simplex die alle kansverdelingen over de drie woorden voorstelt. Elk document is een kansverdeling over woorden en kan voorgesteld worden op deze simplex. Ook de topics zijn een kansverdeling van woorden en kunnen voorgesteld woorden op deze simplex. Elk gegenereerd document is naast een convexe combinatie van alle woorden ook een convexe combinatie van de L topics, waardoor het naast de $(|\mathcal{V}| - 1)$-dimensionale

Woord	Kans
boek	.0127
bladzijde	.0083
schrijver	.0076
boeken	.0076
dichter	.0075
literatuur	.0069
auteur	.0060
literaire	.0060
roman	.0055
schrijvers	.0054
vertaald	.0054
Amsterdam	.0052
uitgeverij	.0050
poëzie	.0049
gedichten	.0045
auteurs	.0045
vertaling	.0043
gedicht	.0042
De	.0041
lezen	.0041

(a) Topic 9

Woord	Kans
Irak	.0117
oorlog	.0082
Verenigde	.0075
Amerikaanse	.0068
VS	.0061
Iraakse	.0056
aanslagen	.0055
Saddam	.0053
Afghanistan	.0052
Staten	.0051
Amerikanen	.0049
Bush	.0047
president	.0047
Naties	.0046
Arabische	.0044
terrorisme	.0044
militaire	.0043
regime	.0039
Hoessein	.0038
Iran	.0038

(b) Topic 53

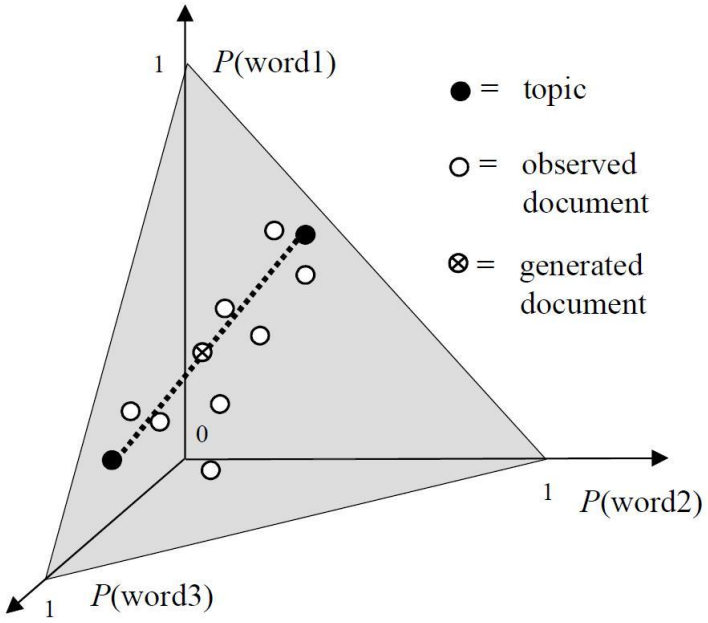
Woord	Kans
televisie	.0142
TV	.0135
VTM	.0086
VRT	.0085
programma	.0073
radio	.0069
programma's	.0069
zender	.0068
kijkers	.0066
omroep	.0060
kranten	.0059
aflevering	.0054
redactie	.0052
kijker	.0047
media	.0046
De	.0045
reportage	.0044
uitgezonden	.0043
canvas	.0042
krant	.0042

(c) Topic 86

Tabel 4.1: Een voorbeeld van de meest waarschijnlijke woorden uit drie topics (van 100 topics in totaal).

simplex ook een punt is van de $(L-1)$-dimensionale simplex gespannen door de topics. In figuur 4.2 is dit te zien. Er zijn twee topics en elk gegenereerd document ligt zowel op de tweedimensionale simplex gespannen door de drie woorden als op de verbindingslijn (eendimensionale simplex) tussen de twee topics.
Als het aantal topics veel kleiner is dan het aantal verschillende woorden ($L << |\mathcal{V}|$), dan spannen de topics een laagdimensionale simplex en de projectie van elk document op deze laagdimensionale simplex kan dan gezien worden als een dimensionaliteitsreductie. Dit is gelijkaardig aan LSA, waar via de afgebroken SWO ook zo een dimensionaliteitsreductie plaatsvindt.

Matrixfactorisatie interpretatie: Naast een geometrische interpretatie heeft het topic model ook een matrixfactorisatie interpretatie gelijkaardig aan LSA. Bij LSA wordt de frequentiematrix W ontbonden via een SWO in drie matrices: een matrix van woordvectoren, een diagonale matrix met singuliere waarden en een matrix met documentvectoren. Deze decompositie is te zien in figuur 4.3. Het topic model kan ook geïnterpreteerd worden als een matrixfactorisatie. Nu is de frequentiematrix W slechts in twee delen opgesplitst: een topic matrix T en een documentmatrix H. Dit is

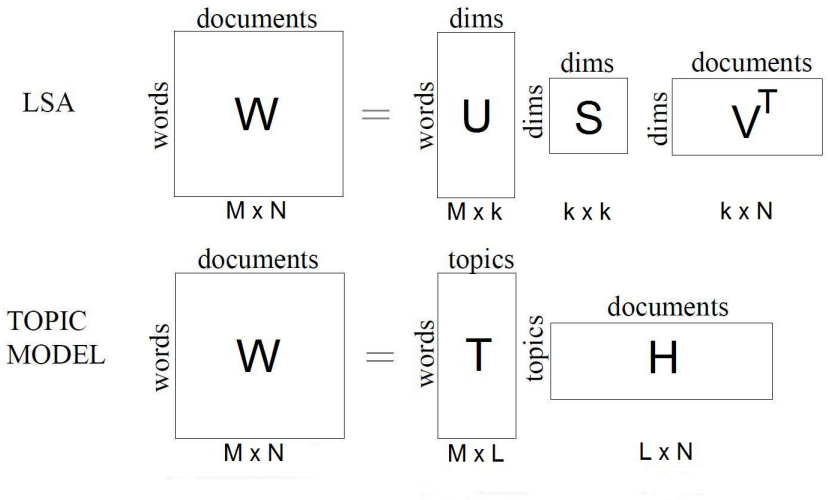


Figuur 4.2: Een geometrische interpretatie van het topic model. Deze figuur komt uit [24].

te zien in figuur 4.3. De overeenkomst tussen de LSA- en het topic modelvoorstelling is nog duidelijker wanneer de diagonaalmatrix S uit LSA opgenomen wordt in de matrix U of V.
Tussen beide decomposities zijn er echter wel nog enkele belangrijke verschillen. Eerst en vooral zijn bij het topic model alle elementen uit T en V niet-negatief en is de som van elke kolom één, terwijl bij LSA de elementen negatief kunnen zijn. Nog een verschil is dat de LSA-decompositie een orthogonale basis voorziet. Dit is computationeel interessant, aangezien een decompositie van dimensie k automatisch de lagere dimensiebenaderingen bevat. Bij topic modellen is er geen orthogonaliteit en moet voor elke dimensiegrootte het model opnieuw uitgerekend worden.

4.3 EM algoritme

De parameters van het topic model zijn a priori niet gekend. Ze moeten uit de trainingsset berekend worden. De trainingsset is hier net als bij LSA de gewogen frequentiematrix W. Als trainingscriterium wordt de log-likelihood $\mathcal{L}$ gebruikt, dit



Figuur 4.3: De matrixfactorisatie van het LSA-model vergeleken met het topic model.

is de log-probabiliteit van de data W onder het model θ:

$$\mathcal{L}(\theta; W) = \sum_{i=1}^{M}\sum_{j=1}^{N} n(w_i, d_j) \log P(w_i, d_j) \tag{4.3}$$

$$= \sum_{i=1}^{M}\sum_{j=1}^{N} n(w_i, d_j) \log \left[P(d_j) \sum_{l=1}^{L} P(w_i|t_l) P(t_l|d_j) \right] \tag{4.4}$$

$$= \sum_{i=1}^{M}\sum_{j=1}^{N} n(w_i, d_j) \log P(d_j) + \sum_{i=1}^{M}\sum_{j=1}^{N} n(w_i, d_j) \log \left[\sum_{l=1}^{L} P(w_i|t_l) P(t_l|d_j) \right] \tag{4.5}$$

hierbij is M het aantal woorden in het vocabularium, N het aantal documenten in de trainingsset en $n(w_i, d_j)$ het gewogen aantal keer dat woord w_i in document d_j voorkomt.
De eerste term in vergelijking 4.5 is onafhankelijk van de tweede en leidt tot een oplossing $P(d_j) \propto n(d_j)$ met $n(d_j)$ het aantal woorden in document d_j [16].
Het aantal topics dient op voorhand vastgelegd te worden. De parameters kunnen gefit worden met behulp van het Expectation-Maximization (EM) algoritme. Als startwaarden krijgen de parameters random waarden toegekend. Er wordt rekening gehouden met de volgende normalisatiebeperkingen:

$$\sum_{i=1}^{M} P(w_i|t_l) = 1 \tag{4.6}$$

voor $l = 1, 2, ..., L$, en

$$\sum_{l=1}^{L} P(t_l|d_j) = 1 \tag{4.7}$$

voor $j = 1, 2, ..., N$.

Het EM-algoritme bestaat uit twee stappen: een E-stap die de posterior probabiliteit berekent van de latente variabelen t_l voor de gegeven parameters, en een M-stap waarin de parameters herschat worden.
De E-stap berekent de kans dat een bepaald woord w_i in een document d_j door een topic t_l is gegenereerd. Met behulp van de regel van Bayes wordt dit [16]:

$$P(t_l|w_i, d_j) = \frac{P(w_i|t_l)P(t_l|d_j)}{\sum_{k=1}^{L} P(w_i|t_k)P(t_k|d_j)} \tag{4.8}$$

De M-stap past de modelparameters aan met de nieuwe waarde van de posterior probabiliteit van de latente variabelen, berekend in de E-stap [16]:

$$P(w_i|t_l) = \frac{\sum_{j=1}^{N} n(w_i, d_j)P(t_l|w_i, d_j)}{\sum_{m=1}^{M}\sum_{j=1}^{N} n(w_m, d_j)P(t_l|w_m, d_j)} \tag{4.9}$$

$$P(t_l|d_j) = \frac{\sum_{i=1}^{M} n(w_i, d_j)P(t_l|w_i, d_j)}{\sum_{k=1}^{L}\sum_{i=1}^{M} n(w_i, d_j)P(t_k|w_i, d_j)} \tag{4.10}$$

De E- en de M-stap wisselen elkaar af tot een stopconditie bereikt is. Mogelijke stopcondities zijn een maximaal aantal iteraties, onvoldoende vooruitgang in de laatste iteraties, een maximale berekeningstijd, ...

4.4 pLSA-taalmodel

4.4.1 Directe modellering

Nu de parameters van het model gekend zijn, kan het pLSA-taalmodel opgesteld worden. De parameters $P(t_l|d_j)$ zijn niet meer nodig, aangezien er nieuwe documenten aan te pas komen. Van deze documenten is echter de topicverdeling onbekend. Deze kunnen geschat worden door de kansen $P(w_i|t_l)$ constant te houden en slechts de vergelijkingen 4.8 en 4.10 te itereren over de voorafgaande woorden in hetzelfde document. In plaats van de volledige berekening voor $P(t_l|H_{i-1})$ uit te voeren tijdens elke stap, kan een online benadering gebruikt worden [15]:

$$P(t_l|H_i) = \frac{1}{i+1}\frac{P(w_i|t_l)P(t_l|H_{i-1})}{\sum_{k=1}^{L} P(w_i|t_k)P(t_k|H_{i-1})} + \frac{i}{i+1}P(t_l|H_{i-1}) \tag{4.11}$$

$$P(t_l|H_1) = P(t_l) = \frac{\sum_{i=1}^{M}\sum_{j=1}^{N} n(w_i, d_j)P(t_l|d_j)}{\sum_{i=1}^{M}\sum_{j=1}^{N} n(w_i, d_j)} \tag{4.12}$$

Eens de topicverdeling over de context $P(t_l|H_{q-1})$ gekend is, kan de kans op voorkomen van het volgende woord bepaald worden via volgende formule:

$$P(w_q|H_{q-1}) = \sum_{l=1}^{L} P(w_q|t_l)P(t_l|H_{q-1}) \tag{4.13}$$

4.4.2 Combinatie met N-grammen

Net als het LSA-taalmodel houdt het pLSA-taalmodel geen rekening met lokale afhankelijkheden, waardoor het een slechte voorspeller is van functiewoorden. Dat zal leiden tot een hoge perplexiteit op een testset. Ook hier kan het taalmodel gecombineerd worden met N-grammen. Dezelfde mogelijkheden tot combinatie als bij LSA zijn mogelijk. Dit zijn de volgende:

1. quasi-Bayesiaanse interpretatie:

$$P_{\text{BAYES}}(w_q|H_{q-1}) = \frac{P_{\text{pLSA}}(w_q|H_{q-1})P_{N\text{-gram}}(w_q|w_{q-1}...w_{q-n+1})}{P(w_q)\sum_{w_i\in\mathcal{V}} \frac{P_{\text{pLSA}}(w_i|H_{q-1})}{P(w_i)} P_{N\text{-gram}}(w_i|w_{q-1}...w_{q-n+1})} \tag{4.14}$$

2. lineaire interpolatie:

$$P_{\text{LI}}(w_q|H_{q-1}) = \lambda P_{\text{pLSA}}(w_q|H_{q-1}) + (1-\lambda)P_{N\text{-gram}}(w_q|w_{q-1}...w_{q-n+1}) \tag{4.15}$$

3. *information weighted arithmetic mean:*

$$P_{\text{IWAM}}(w_q|H_{q-1}) = \frac{\lambda_q P_{\text{pLSA}}(w_q|H_{q-1}) + (1-\lambda_q)P_{N\text{-gram}}(w_q|w_{q-1}...w_{q-n+1})}{\sum_{w_i\in\mathcal{V}} \lambda_i P_{\text{pLSA}}(w_i|H_{q-1}) + (1-\lambda_i)P_{N\text{-gram}}(w_i|w_{q-1}...w_{q-n+1})} \tag{4.16}$$

4. *information weighted geometric mean:*

$$P_{\text{IWGM}}(w_q|H_{q-1}) = \frac{P^{\lambda_q}_{\text{pLSA}}(w_q|H_{q-1})P^{(1-\lambda_q)}_{N\text{-gram}}(w_q|w_{q-1}...w_{q-n+1})}{\sum_{w_i\in\mathcal{V}} P^{\lambda_i}_{\text{pLSA}}(w_i|H_{q-1})P^{(1-\lambda_i)}_{N\text{-gram}}(w_i|w_{q-1}...w_{q-n+1})} \tag{4.17}$$

5. *similarity modulated N-gram:*

$$P_{\text{SIMMOD}}(w_q|H_{q-1}) = \frac{P_{\text{pLSA}}(w_q|H_{q-1})P_{N\text{-gram}}(w_q|w_{q-1}...w_{q-n+1})}{\sum_{w_i\in\mathcal{V}} P_{\text{pLSA}}(w_i|H_{q-1})P_{N\text{-gram}}(w_i|w_{q-1}...w_{q-n+1})} \tag{4.18}$$

4.5 Discussie

4.5.1 Verband met LSA

Er zijn heel wat overeenkomsten tussen pLSA en LSA. Eerder werd al aangegeven dat er bij beide modellen een dimensionaliteitsreductie plaatsvindt. De woorden en documenten worden getransformeerd naar een lage dimensie waarin ze onderling te vergelijken zijn. Ook werd al vermeld dat beide modellen een matrixfactorisatie uitvoeren op de frequentiematrix W. LSA ontbindt deze matrix in drie delen, terwijl pLSA de matrix W ontbindt in twee delen.

Een cruciaal verschil tussen LSA en pLSA is de doelfunctie, gebruikt bij de optimale decompositie. Bij LSA is de doelfunctie een Frobeniusnorm, waardoor verondersteld wordt dat de ruis in de frequentiematrix Gaussisch verdeeld is. Deze veronderstelling is niet nodig bij pLSA, waar de doelfunctie een likelihood functie van samples is, om een model te verkrijgen met de sterkste voorspellende kracht.
Een tweede verschil met LSA is dat pLSA het probleem van polysemie oplost. In topic modellen wordt er een onderscheid gemaakt tussen de verschillende betekenissen van eenzelfde woord. Een voorbeeld hiervan is het woord *stem*. Dit kan zowel de menselijke stem zijn als een verkiezingsstem. In tabel 4.2 zijn de meest voorkomende woorden uit twee topics van 100 gegeven. Het pLSA-model is getraind op data uit De Standaard. In topic 39 wijst *stem* op de menselijke zangstem, terwijl in topic 50 *stem* verwijst naar een verkiezingsstem. In een document over bv. muziek zal *stem* dan steeds uit topic 39 gekozen worden. Een document over de verkiezingsstrijd in Frankrijk zal het woord *stem* uit topic 50 halen.
Een ander verschil van pLSA t.o.v. LSA heeft te maken met het verkregen model. Bij pLSA hebben de elementen in het model een eigen probabilistische betekenis. Ook zijn de kolommen genormaliseerd. De elementen uit het LSA model hebben geen eigen betekenis, ze kunnen zelfs negatief zijn. De kolommen of rijen zijn ook niet genormaliseerd. De verschillende dimensies van de gereduceerde LSA-ruimte zijn eveneens niet interpreteerbaar, terwijl de dimensies uit het pLSA-model, de topics, wel een eigen interpretatie kunnen hebben, zie eerder in dit hoofdstuk. Het voordeel van een probabilistisch model is dat er standaardtechnieken uit statistiek kunnen gebruikt worden voor modelselectie en complexiteitscontrole.
Het pLSA-model is niet op alle vlakken beter dan het LSA-model. Een belangrijk nadeel van het EM-algoritme is dat het slechts een lokaal optimum vindt i.p.v. het gewenste globale optimum. De SWO daarentegen kan exact berekend worden wat de garantie biedt dat de oplossing van het LSA-model wel een globaal optimum is.

Een belangrijke zwakheid die bij pLSA overblijft is de rekemcomplexiteit. Die is net als bij LSA hoog. Het EM-algoritme vraagt meerdere iteraties en neemt veel tijd in beslag. Daarbij komt kijken dat de oplossing slechts een lokaal optimum is en bijgevolg slecht kan zijn. Hierdoor zijn dikwijls meerdere pogingen nodig vanuit verschillende startoplossingen. Eens het model gekend is, moeten voor elk te voorspellen woord de kansen $P(t_l|H_{q-1})$ berekend worden. Dit vraagt meer werk

dan de snelle opzoekingen bij het N-gram taalmodel.
Het probleem van de woordvolgorde, waarmee het pLSA-taalmodel geen rekening houdt, wordt net als bij LSA opgelost door het taalmodel te combineren met N-grammen.

Woord	Kans
muziek	.0070
plaat	.0051
jazz	.0041
songs	.0037
album	.0035
CD	.0034
gitaar	.0029
muzikanten	.0028
band	.0023
zingt	.0023
nummers	.0022
blues	.0021
strips	.0021
trio	.0020
albums	.0020
strip	.0019
muzikale	.0019
stem	.0019
drummer	.0018
folk	.0017

(a) Topic 39

Woord	Kans
stemmen	.0130
verkiezingen	.0097
partijen	.0090
politieke	.0076
kiezers	.0070
partij	.0066
politiek	.0062
politici	.0060
meerderheid	.0058
macht	.0043
stem	.0041
beleid	.0039
kandidaten	.0038
kandidaat	.0035
democratie	.0035
stemt	.0033
verkiezing	.0032
gestemd	.0032
stemming	.0031
referendum	.0030

(b) Topic 50

Tabel 4.2: Twee topics die het woord *stem* bevatten in een andere betekenis. In topic 39 wijst *stem* op de zangstem, en in topic 50 is *stem* een verkiezingsstem. Het pLSA-model is getraind op data uit De Standaard.

4.5.2 Topics versus clusters

Op het eerste gezicht lijken topics op clusters. Er zijn echter wel duidelijke verschillen tussen beide. In clusteringmodellen is elk document geassocieerd met één latente variabele. In een probabilistisch clusteringmodel is de woordkans gedefinieerd als [16]:

$$P(w_i|d_j) = \sum_{k=1}^{K} P\{c(d_j) = c_k\} P(w_i|c_k) \tag{4.19}$$

met $P\{c(d_j) = c_k\}$ de posteriorkans dat document d_j tot cluster c_k behoort.

Deze vergelijking is algebraisch identiek aan vergelijking 4.2, maar toch zijn ze conceptueel niet hetzelfde. In een clusteringmodel wordt ervan uitgegaan dat elk

document tot exact één cluster behoort. Bij het topic model daarentegen is slechts het optreden van één woord verbonden aan een uniek topic t_l. Een document kan woorden bevatten die uit verschillende topics gekozen zijn. Ook kan een zelfde woord in verschillende documenten gekozen zijn uit verschillende topics. Elk document is in het topic model bijgevolg een convexe combinatie van de topics.

4.6 pLSA als niet-negatieve matrixfactorisatie

Een niet-negatieve matrixfactorisatie (NMF) ontbindt een matrix W in een product van niet-negatieve factoren:

$$W \approx T.H \tag{4.20}$$

Het spreekt voor zich dat alle elementen uit W positief moeten zijn. De meest gebruikte doelfuncties die de kwaliteit van benadering kwantificeren zijn de Euclidische afstand en de Kullback-Leibler (KL) divergentie [21]. De KL-divergentie tussen A en B is gedefinieerd als:

$$D(A||B) = \sum_{ij} \left(A_{ij} \log \frac{A_{ij}}{B_{ij}} - A_{ij} + B_{ij} \right) \tag{4.21}$$

De KL-divergentie $D(A||B)$ is niet-dalend bij gebruik van volgende updateoperaties [21]:

$$H_{lj} \leftarrow H_{lj} \frac{\sum_i \frac{T_{il} V_{ij}}{(TH)_{ij}}}{\sum_i T_{il}}, \quad T_{il} \leftarrow T_{il} \frac{\sum_j \frac{H_{lj} V_{ij}}{(TH)_{ij}}}{\sum_j H_{lj}} \tag{4.22}$$

In [9] en [14] wordt aangetoond dat er een equivalentie is tussen de niet-negatieve matrixontbinding en de ontbinding aanwezig bij pLSA. De redenering gaat als volgt:

Aanschouw een equivalente parametrisatie van het probabilistisch topic model [16]:

$$P(w_i, d_j) = \sum_{l=1}^{L} P(t_l) P(d_j|t_l) P(w_i|t_l) \tag{4.23}$$

Introduceer een $M \times L$ matrix T' , zodat $T'_{il} = P(w_i|t_l)P(t_l)$, en een $L \times N$ matrix H', zodat $H'_{lj} = P(d_j|t_l)$. Dan geldt dat $[P(w_i, d_j)] = T'H'$.
Aanschouw terug vergelijking 4.20. Veronderstel zonder verlies aan algemeenheid dat $\sum_{ij} W_{ij} = 1$. Introduceer de matrices A en B, twee $L \times L$ diagonale matrices zodanig dat $A_{LL} = \sum_i T_{iL}$ en $B_{LL} = \sum_j H_{Lj}$. Dan geldt:

$$T.H = (T.A^{-1}.A) \times (B.B^{-1}.H) = (T.A^{-1}) \times (A.B) \times (B^{-1}.H) \tag{4.24}$$

waarbij $T.A^{-1}$ en $B^{-1}.H$ de eigenschappen bezitten van een voorwaardelijke probabiliteitsmatrix $[P(w_i|t_l)]$ respectievelijk $[P(d_j|t_l)]^T$, en $A.B$ is gerelateerd aan een diagonaalmatrix van $P(t_l)$. Dit toont aan dat NMF zeer gerelateerd is aan pLSA.

In [14] wordt nu bewezen dat een lokale maximum likelihood oplossing van pLSA eveneens een oplossing is van het NMF probleem met KL-divergentie. Een vast punt van het EM-algoritme zal ook een vast punt zijn van de updateoperaties uit 4.22. In [9] tonen ze dan weer aan dat de doelfunctie van pLSA equivalent is met de doelfunctie van NMF met KL-divergentie.
Bijgevolg geldt dat beide methodes een zelfde probleem oplossen, maar dat ze elk via een ander algoritme een oplossing vinden. Dit wilt zeggen dat een lokale oplossing van het pLSA-model naast het EM-algoritme ook gevonden kan worden met de updateoperaties uit vergelijking 4.22.

Hoofdstuk 5

Experimenten

De verschillende taalmodellen staan uitvoerig beschreven in de vorige hoofdstukken. In dit hoofdstuk zal aan de hand van experimenten uitgemaakt worden welk model het best geschikt is voor taalmodellering. In het begin wordt telkens het model aangemaakt met de data uit de trainingsset. Vervolgens vindt het afstellen van de onbekende parameters plaats, dit met behulp van de validatieset. In de laatste stap wordt het model uitgevoerd op twee onafhankelijke testsets en wordt de perplexiteit per taalmodel berekend. Op basis van deze perplexiteit kunnen de verschillende taalmodellen met elkaar vergeleken worden.

5.1 Opbouw

5.1.1 Data

De eerste belangrijke keuze die moet gemaakt worden, is de taal waarop de modellen zullen getest worden. In deze masterproef is gekozen om te experimenteren op Nederlandstalige data. De Mediargusdata uit de periode 1999-2004 is beschikbaar als tekstdata. Deze data bevat elke uitgave van de belangrijkste Vlaamse en Nederlandse kranten en tijdschriften uit die periode. Per krant of tijdschrift is ook een genormaliseerde versie van deze data beschikbaar. In deze versie zijn onder andere alle leestekens verwijderd, enkel het begin en het einde van elke zin is aangegeven. Het enige nadeel van deze versie is dat de data niet in afzonderlijke documenten is verdeeld. In de experimenten zal steeds de genormaliseerde versie van de data gebruikt worden.

De volledige verzameling van De Standaard uit de periode 1999-2004 vormt de trainingsset in de experimenten. In totaal bevat deze verzameling 65 miljoen woorden. Hiermee zal dan het N-gram, LSA- en pLSA-taalmodel getraind worden.
Zowel bij het caching-, LSA- als het pLSA-taalmodel zijn er nog enkele parameters die moeten afgesteld worden. Deze parameters kunnen experimenteel bepaald worden door het minimaliseren van de perplexiteit op een validatieset. Ter herinnering, de

perplexiteit wordt als volgt berekend:

$$PP = 10^{-\frac{1}{Q}\sum_{q=1}^{Q}\log_{10} P(w_q|H_{q-1})} \tag{5.1}$$

waarbij Q de grootte is van de validatieset en 10 als basis van het logaritme gekozen is. In de macht van 10 staat de gemiddelde log-probabiliteit over de validatieset. Let ook op dat de waarde van de perplexiteit onafhankelijk is van de keuze van basis van het logaritme. De validatieset bestaat in deze experimenten uit 100 000 woorden uit De Morgen, ook uit de periode 1999-2004.
Ten slotte bestaat er de mogelijkheid om de verschillende taalmodellen met elkaar te vergelijken. Dit kan door hun perplexiteit op onafhankelijke testsets te berekenen. De perplexiteit wordt op dezelfde manier als bij de validatieset gevonden. De eerste testset bestaat uit 50 000 woorden uit Knack, de tweede testset bevat 50 000 woorden uit de Nederlandse krant NRC Handelsblad. De tekstdata komt eveneens uit de periode 1999-2004.

5.1.2 Vocabularium

Een belangrijke ontwerpbeslissing die nog moet gemaakt worden is de keuze van het vocabularium $\mathcal{V}$. Een mogelijkheid is om elk uniek woord uit de trainingsset op te nemen in het vocabularium. Dit is echter geen verstandige keuze, aangezien er dan heel wat woorden in zullen zitten die slechts een weinig aantal keer zijn voorgekomen in de trainingsset, zoals bv. *Takashimaya-grootwarenhuis*. De kans is klein dat deze woorden hierna nog zullen voorkomen in de validatie- of testset. Een beter idee is om te werken met een beperkt vocabularium, waarin alleen de woorden zitten die het meest frequent voorkomen. Zodoende moet er een afweging gemaakt worden tussen de grootte van het vocabularium en de out-of-vocabulary (OOV) rate, dit is het aantal woorden uit de testset die niet in het vocabularium voorkomen. Een kleiner vocabularium zal leiden tot een sterkere voorspelling van de woorden uit het vocabularium, maar zorgt ook voor een hogere OOV-rate. In [3] en [8] is telkens een vocabularium van ongeveer 20 000 woorden gebruikt. De experimenten werden wel uitgevoerd op Engelstalige data. Het verschil in vocabularium tussen Engels en Nederlands ligt voornamelijk bij de samenstellingen. Deze worden in het Engels dikwijls in twee woorden geschreven. Deze woorden kunnen afzonderlijk reeds in het vocabularium staan. In het Nederlands worden samenstellingen echter aan elkaar geschreven, wat leidt tot een nieuw woord dat nog niet in het vocabularium voorkomt. Bijgevolg zal het vocabularium in een Nederlandse spraakproef groter moeten zijn dan het vocabularium gebruikt voor een Engelse spraakproef. In [22] wordt aangetoond dat voor een zelfde OOV-rate het Nederlandse vocabularium 65 000 woorden groot moet zijn, tegenover een Engels vocabularium van 20 000 woorden. Na enkele tests bleek dat een woordenboek van meer dan 100 000 woorden nodig was om de OOV-rate te beperken tot minder dan 5% in de validatie- en testsets. In deze experimenten bestaat het vocabularium uit de 105 000 meest frequente woorden uit de trainingsset. Dit leidt tot een OOV-rate van 3.3% in de validatieset en een OOV-rate van 3.0% in de eerste en 4.3% in de tweede testset.

5.2 N-grammen

De N-gram taalmodellen zijn gecreëerd en getest met behulp van SRILM, een collectie van C++ bibliotheken [25]. Deze toolkit laat toe om naast de trainingsset en een waarde voor N een zelfgekozen vocabularium mee als parameter op te geven. De gewenste smoothingtechniek kan ook als parameter meegegeven worden, net als de keuze tussen backoff en interpolatie voor lagere orde tellingen. De mogelijke smoothingtechnieken zijn additieve smoothing, Witten-Bell discounting, Good-Turing discounting, absolute discounting, en de gewijzigde Kneser-Ney smoothingtechniek. Standaard wordt er backoff toegepast en is Good-Turing discounting de gebruikte smoothingtechniek. Eens het model berekend is kan de perplexiteit op een testset berekend worden. De testset en het model worden als parameter meegegeven. Met behulp van de validatieset zal uitgemaakt worden welke parameterwaarden het best geschikt zijn voor het N-gram taalmodel.

In het eerste experiment wordt N gevarieerd en krijgen de andere parameters de standaardwaarden. N is hierbij omwille van zowel computationele als sparsiteitsredenen beperkt gehouden tot maximaal 5. In tabel 5.1 zijn de resultaten op de validatieset te zien. Het valt op dat er grote winst geboekt wordt wanneer N van 2 naar 3 gaat, en matige winst wanneer N van 3 naar 4 gaat. N verder verhogen tot 5 levert nauwelijks een beter resultaat op.

Gebruikt model	**Perplexiteit**
2-gram	417.10
3-gram	298.46
4-gram	277.41
5-gram	276.10

Tabel 5.1: Perplexiteitresultaten voor de verschillende ordes in het N-gram taalmodel.

In het volgende experiment wordt de orde N constant gehouden op 3 en wordt de invloed van de gebruikte smoothingtechniek getest, samen met backoff of interpolatie. Niet alle combinaties van smoothingtechniek en backoff/interpolatie zijn echter mogelijk in SRILM. In tabel 5.2 zijn de resultaten weergegeven voor de mogelijke combinaties. Zoals verwacht zijn de resultaten slecht voor additieve smoothing. Voor de andere smoothingtechnieken liggen de resultaten niet ver uit elkaar. Net zoals in [5] scoort de gewijzigde Kneser-Ney smoothingtechniek beter dan de andere smoothingtechnieken. In combinatie met interpolatie scoort het net iets beter dan in combinatie met backoff. Als definitief N-gram taalmodel wordt een 3-gram gekozen met gewijzigde Kneser-Ney smoothing en interpolatie. In de combinatie met andere taalmodellen zal ook dit model gebruikt worden. N is bewust laag gehouden op 3, opdat de context zeer lokaal blijft bij het N-gram. De andere taalmodellen zullen immers meer met de globale context rekening houden.

Smoothingtechniek	Backoff/Interpolatie	Perplexiteit
add-one	backoff	2464.04
add-0.01	backoff	415.58
Good-Turing	backoff	298.46
Witten-Bell	backoff	296.49
	interpolatie	302.66
absolute discounting	backoff	299.93
	interpolatie	318.79
gewijzigde Kneser-Ney	backoff	294.82
	interpolatie	**293.72**

Tabel 5.2: Perplexiteitresultaten voor de verschillende smoothingtechnieken en met backoff/interpolatie in het N-gram taalmodel.

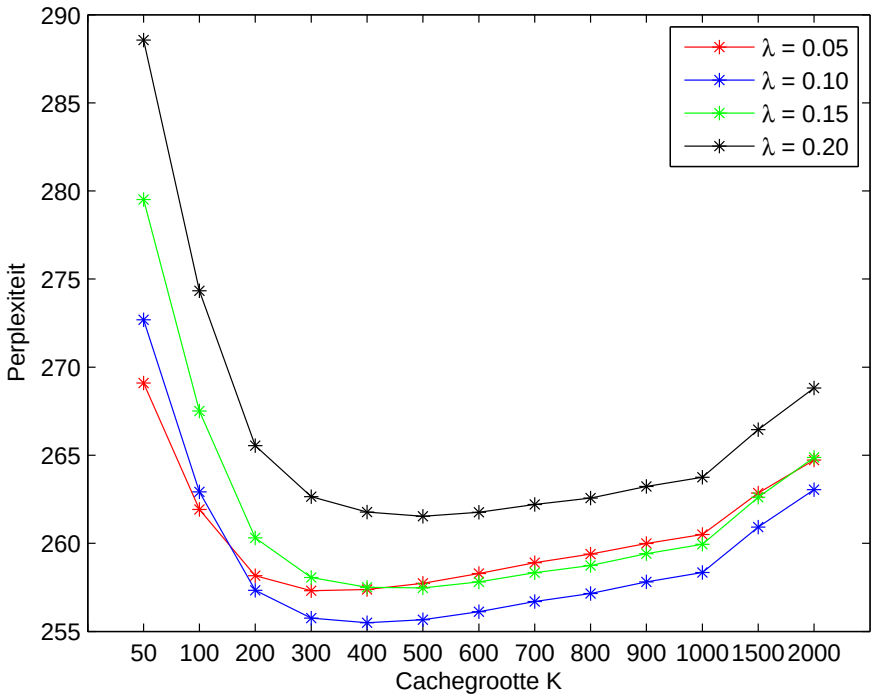
5.3 Cachingmodellen

Het opstellen van cachingmodellen vereist geen trainingsfase. Cachingmodellen zullen enkel de voorgaande herkende woorden gebruiken in het model. Dit betekent dat ze slechts bruikbaar zijn als er al een deel tekst herkend is. Voor de herkenning van dit eerste deel is een ander taalmodel vereist. In dit experiment wordt er vanuit gegaan dat dit eerste stuk tekst al herkend is.
Cachingmodellen zijn als alleenstaand model niet voldoende betrouwbaar en worden bij voorkeur gecombineerd met een ander taalmodel. In dit experiment worden ze gecombineerd met N-grammen via een lineaire interpolatie, zie vergelijking 2.21. Om een gecombineerd taalmodel te verkrijgen zijn nog twee parameters te bepalen, namelijk de cachegrootte K en de interpolatieparameter λ. Voor een exponentieel cachingmodel is ook nog de afnamefactor α onbekend. De optimale waarden voor deze parameters kunnen gevonden worden door het minimaliseren van de perplexiteit op de validatieset.

5.3.1 Uniforme cache

Eerst wordt de invloed van de cachegrootte K en interpolatieparameter λ getest voor een uniforme cache. In figuur 5.1 staan de perplexiteitwaarden uitgezet voor verschillende waarden van K en λ. Een kleine waarde voor de interpolatieparameter λ is gewenst, daar het aandeel van de betrouwbaardere N-gram kans groot genoeg zou zijn. De beste resultaten zijn duidelijk voor $\lambda = 0.1$. De ideale grootte van de cache is moeilijk te voorspellen. Waarschijnlijk zal deze niet groter zijn dan het aantal woorden in een document, aangezien caching modellen voornamelijk geschikt zijn om woorden te voorspellen binnen éénzelfde document. De cachegrootte mag ook niet te laag liggen, want dan is er te weinig informatie beschikbaar om te modelleren. Uit de resultaten blijkt dat de optimale cachegrootte K rond 400 ligt. Het verschil met $K = 300$ of $K = 500$ is echter niet groot. Dit resultaat is in overeenstemming

met [6], waar de optimale cachegrootte 500 bedraagt.
In het finale model wordt er gekozen voor een interpolatieparameter $\lambda = 0.1$ en een cachegrootte $K = 400$.

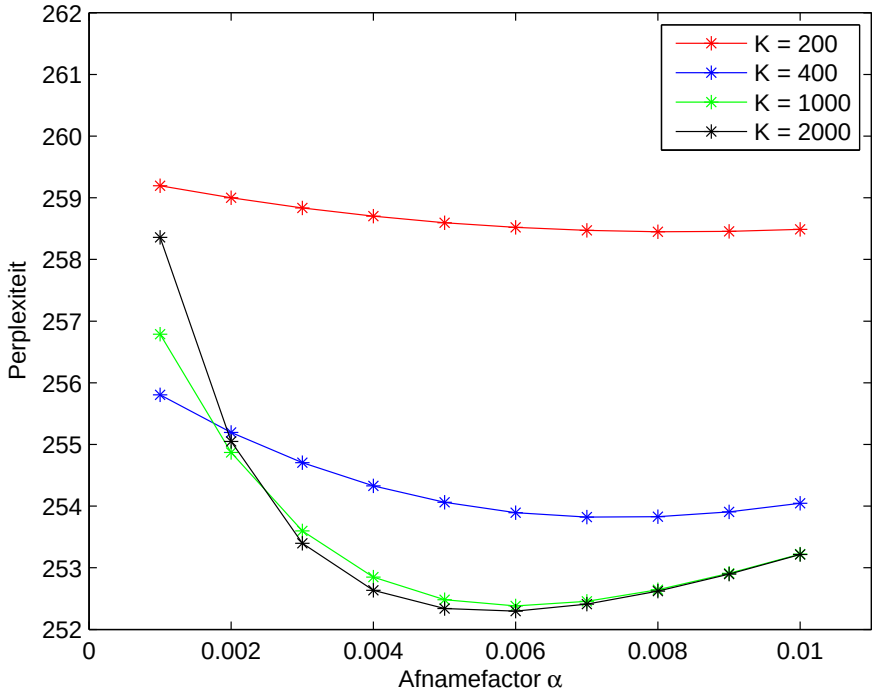


Figuur 5.1: Invloed van de cachegrootte K en interpolatieparameter λ bij een uniform cachingmodel.

5.3.2 Exponentiële cache

Een exponentiële cache heeft nog een bijkomende parameter: de afnamefactor α. Deze kan eveneens experimenteel bepaald worden. Het volgend experiment test de invloed van de afnamefactor α en de cachegrootte K. Om bijkomende tests te vermijden, is verondersteld dat de beste waarde voor de interpolatieparameter $\lambda = 0.1$ is, het beste resultaat verkregen bij een uniforme cache.
De cachegrootte kan hier ook onbeperkt zijn, dit wilt zeggen dat alle woorden vanaf het begin in rekening worden gebracht. Door de exponentiële afname zal hun aandeel echter snel zakken naarmate de tekst vordert. Aangezien steeds de hele cache doorlopen wordt, is deze aanpak computationeel veel intensiever dan bij een beperkte cachegrootte. Dit geval wordt dan ook niet verder bestudeerd.
De resultaten van de tests met verschillende waarden voor de cachegrootte K en afnamefactor α zijn te zien in figuur 5.2. Het minimum wordt voor elke cachegrootte ongeveer bereikt voor $\alpha = 0.006$. Dit ligt dicht bij het beste resultaat in [6], nl.

$\alpha = 0.005$. In tegenstelling tot bij een uniforme cache is nu een grotere cache gewenst. De perplexiteit ligt beduidend lager voor $K = 1000$ of 2000 dan voor $K = 400$. Aangezien het verschil tussen $K = 1000$ en $K = 2000$ haast verwaarloosbaar klein is en een kleinere cachegrootte computationeel gunstiger is, wordt de cachegrootte $K = 1000$ verkozen. In het finale model is $K = 1000$, $\alpha = 0.006$ en $\lambda = 0.1$.



Figuur 5.2: Invloed van de cachegrootte K en afnamefactor α bij een exponentieel cachingmodel.

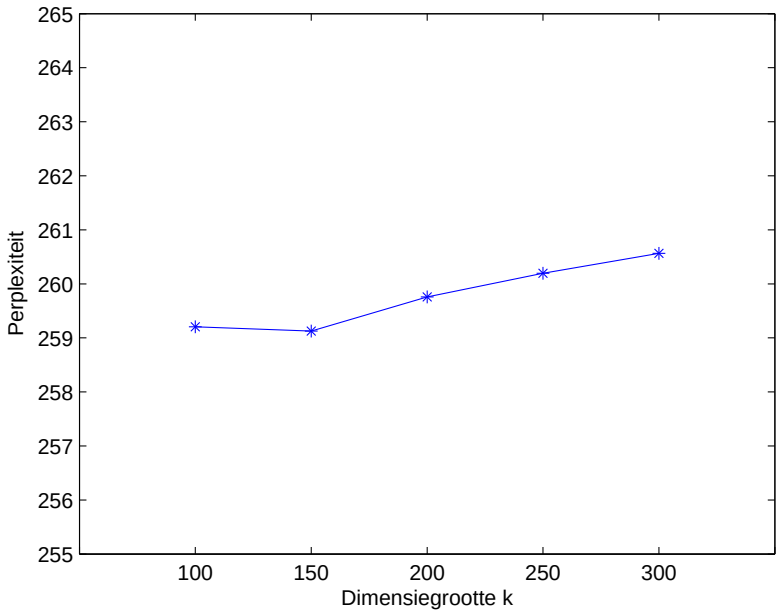
5.4 LSA

Om LSA optimaal te kunnen toepassen, moet de data opgesplitst zijn in afzonderlijke documenten. De beschikbare data bestaat uit een aaneenschakeling van documenten, echter zonder markeringen tussen twee opeenvolgende documenten. De traditionele opsplitsing in documenten is bijgevolg niet mogelijk en zodoende moet er een alternatieve opsplitsing gebruikt worden. Een mogelijkheid is om de data op te splitsen in even grote stukken, waarbij elk stuk ongeveer even lang is als een gemiddeld document. In dat geval heeft elk document een even grote invloed en moet er niet genormaliseerd worden op documentlengte. In de volgende experimenten werd de data opgesplitst per 20 zinnen. De eerste 20 zinnen vormen zo het eerste document, de volgende 20 zinnen het tweede document, ... Het nadeel van deze aanpak is dat de oorspronkelijk kleine documenten steeds gemengd zullen zijn met

andere documenten en dat de oorspronkelijk langere documenten opgesplitst zullen zijn over meerdere nieuwe documenten. Het voordeel is dat elk document evenveel zinnen bevat en bijgevolg ook ongeveer evenveel woorden. Op deze manier wordt de trainingsset opgesplitst in een kleine 400 000 documenten. Aangezien het aanmaken van het LSA-model computationeel veel werk vraagt, zijn geen andere waarden dan 20 getest. Uit eerste tests bleek echter dat de resultaten positief waren voor deze waarde. Een alternatief is om een sliding window te gebruiken. In dat geval overlappen opeenvolgende documenten deels met elkaar. In deze masterproef is dit niet verder onderzocht, maar dit kan wel een aanzet zijn tot verder onderzoek.
Eens de onderverdeling in documenten gemaakt is, kan de frequentiematrix opgesteld worden. Enkel tellingen van woorden die uit het vocabularium $\mathcal{V}$ komen, worden in de matrix opgenomen. De elementen worden dan gewogen volgens de methode beschreven in vergelijkingen 3.1-3.5. Andere wegingsmethoden zijn mogelijk, maar werden hier niet getest. Vervolgens kan het LSA-model opgesteld worden via de afgebroken SWO. De rang k van de gereduceerde matrix is nog onbekend en moet experimenteel bepaald worden.
Dan kunnen met de verkregen matrices U en S de kansen bepaald worden van de woorden uit de validatie- en testset. Een nieuw probleem dat hier optreedt is dat nu normaal de afstand tussen het nieuwe woord en de documentgeschiedenis wordt berekend. Omdat ook de validatie- en testset niet opgesplitst zijn in afzonderlijke documenten, wordt deze aanpak aangepast. In plaats van de afstand tussen het volgende woord en de documentgeschiedenis te berekenen, wordt de afstand tussen het volgende woord en een aantal vorige woorden berekend. De lengte van deze woordgeschiedenis is een parameter die ook experimenteel zal bepaald worden. Deze aanpak maakt het onnodig om de validatie- en testset in documenten onder te verdelen. Het nadeel is wel dat de eerste woorden van een oorspronkelijk document voornamelijk zullen voorspeld worden door woorden uit een vorig document.
Voorts zijn er nog twee andere parameters die experimenteel moeten bepaald worden: de exponent γ uit vergelijking 3.31 en de interpolatieparameter λ gebruikt bij lineaire interpolatie in vergelijking 3.41. In elke van de volgende tests wordt de perplexiteit op de validatieset vergeleken voor de verschillende parameterwaarden. De eerste drie tests maken gebruik van het gecombineerde model IWGM tussen LSA en een 3-gram, zie vergelijking 3.45. De vierde test maakt gebruik van een lineaire interpolatie tussen de N-gram en LSA-kans om de optimale waarde van λ te vinden. Tenslotte worden in de laatste test alle combinatiemodellen tussen LSA en N-grammen vergeleken, zoals beschreven in vergelijkingen 3.40,3.41 en 3.44-3.46.
Aangezien het alleenstaande LSA-model niet tot competitieve resultaten leidt, zijn deze resultaten niet weergegeven.

5.4.1 Dimensiegrootte k

LSA-modellen met dimensie $k = 100, 150, 200, 250$ en 300 zijn getest op de validatieset. De lengte van de woordgeschiedenis is hier op 50 gezet, deze waarde is gekozen na enkele korte tests. De waarde van de exponent $\gamma = 6$, wat ongeveer overeenkomt met de gesuggereerde waarde uit de literatuur. In figuur 5.3 staan de resultaten van



Figuur 5.3: Invloed van de dimensiegrootte k op het LSA-model.

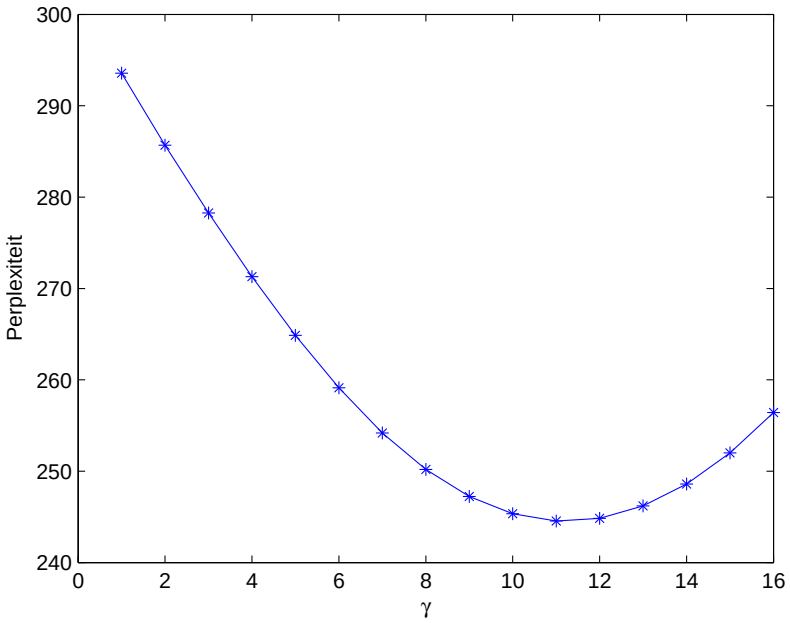
de perplexiteit voor de verschillende dimensiegroottes. Het valt op dat het verschil tussen de verschillende dimensiegroottes klein is. Het minimum wordt bereikt voor $k = 150$. Dit ligt in de lijn van [3], waar een dimensiegrootte $k = 125$ wordt gesuggereerd. In het finale LSA-model wordt $k = 150$ gekozen.

5.4.2 Exponent γ

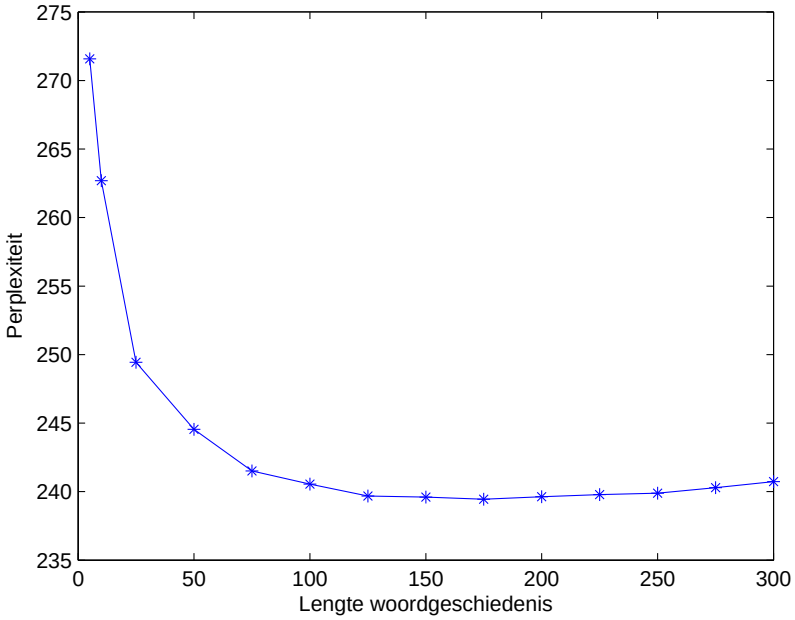
De resultaten van de tests van de exponent γ staan in figuur 5.4. Bij deze tests is de dimensiegrootte $k = 100$ en is de lengte van de woordgeschiedenis 50. De perplexiteit bereikt een minimum voor $\gamma = 11$, deze waarde wordt gebruikt in het finale LSA-model. Dit verschilt met het optimale resultaat uit [8] en [23], waar γ telkens kleiner is dan 10.

5.4.3 Lengte woordgeschiedenis

De lengte van de woordgeschiedenis is getest voor waarden tussen 5 en 300. In figuur 5.5 zijn de resultaten weergegeven. De andere gebruikte parameters zijn $k = 150$ en $\gamma = 11$. Het beste resultaat wordt bekomen voor een lengte van 175 woorden. Een kleine verlaging of verhoging van het aantal woorden heeft slechts een kleine invloed op de perplexiteit. In het finale LSA-model zal ook een lengte van 175 woorden gebruikt worden.



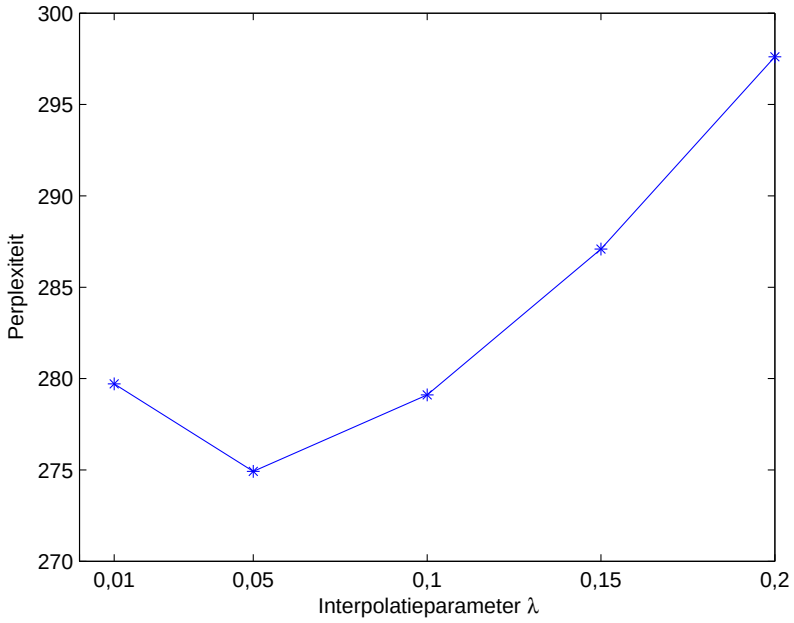
Figuur 5.4: Invloed van de exponent γ op het LSA-model.



Figuur 5.5: Invloed van de lengte van de woordgeschiedenis op het LSA-model.

5.4.4 Interpolatieparameter λ

De invloed van de interpolatieparameter λ, die gebruikt wordt bij de lineaire interpolatie tussen het N-gram en LSA-model, is getest en het resultaat is te zien in figuur 5.6. Hier is een lagere waarde dan bij de combinatie tussen N-gram en cachingmodellen beter. De optimale waarde is $\lambda = 0.05$.



Figuur 5.6: Invloed van de interpolatieparameter λ op het gecombineerde N-gram en LSA-model.

5.4.5 Combinatiemodellen met N-grammen

In tabel 5.3 staan de perplexiteitresultaten voor de verschillende combinatiemodellen tussen N-grammen en LSA. Bij lineaire interpolatie is de interpolatieparameter $\lambda = 0.05$.
Het resultaat van het model met de quasi-Bayesiaanse interpretatie is desastreus. Ook het gelijkaardige model SIMMOD scoort slecht. De modellen die gebruik maken van de betrouwbaarheidsfactor λ_q scoren heel wat beter. Zoals verwacht scoort de niet-lineaire combinatie tussen de N-gram en LSA-kans, IWGM, het best. Het tweede beste model is de eenvoudige lineaire interpolatie, voor IWAM. In het finale LSA-model zal het combinatiemodel IWGM gebruikt worden.

Combinatiemodel	Perplexiteit
quasi-Bayes	4138.69
LI	272.56
IWAM	300.22
IWGM	**239.45**
SIMMOD	982.05

Tabel 5.3: Vergelijking van de combinatiemodellen tussen N-grammen en LSA.

5.5 pLSA

Net als LSA gebruikt pLSA de frequentiematrix W om het taalmodel op te stellen. Aangezien voor het opstellen van deze matrix het trainingscorpus in afzonderlijke documenten moet opgesplitst zijn, wordt ook hier het corpus opgedeeld in stukken van 20 zinnen.
De logische manier om de parameters van het model te berekenen is met behulp van het EM-algoritme, beschreven in vergelijkingen 4.8-4.10. Nu treedt er echter een geheugenprobleem op. Om alle termen $P(t_l|w_i, d_j)$ uit vergelijking 4.10 te berekenen, zijn er $\#\text{topics} \times |\mathcal{V}| \times \#\text{documenten} \approx \#\text{topics} \times 100000 \times 400000$ variabelen nodig, ofwel 4×10^{10} variabelen per topic. Indien de variabelen in enkele precisie worden berekend en er slechts 10 topics in het model zijn, komt dit neer op $4 \times 10 \times 4 \times 10^{10}$bytes = 1.6 TB geheugenopslag. Dit is te veel geheugen voor de beschikbare computers in dit experiment. Het EM-algoritme kan zodoende niet gebruikt worden om het model op te stellen. Als alternatief om het model op te stellen kan de niet-negatieve matrixfactorisatie gebruikt worden, beschreven in sectie 4.6. Hierbij is minder geheugenopslag vereist. Enkel de matrices T en H moeten bewaard worden, wat neerkomt op ongeveer 100 000 variabelen, respectievelijk 400 000 variabelen per topic. In totaal komt dit op ongeveer 500 000 variabelen per topic, heel wat minder dan de 4×10^{10} variabelen bij het EM-algoritme. Aangezien het geheugen hiervoor groot genoeg is, kan het pLSA-model berekend worden via de NMF. Een parameter die nog experimenteel bepaald moet worden is het aantal topics L.
Na het berekenen van de modelparameters kunnen de kansen op de woorden uit de validatie- en testset bepaald worden met formule 4.13. Om deze formule te kunnen gebruiken, moeten eerst de kansen $P(t_l|H_{q-1})$ berekend worden. Aangezien de datasets oorspronkelijk niet in documenten verdeeld zijn en bijgevolg de context niet eenduidig bepaald is, kunnen de formules 4.11-4.12 niet meteen gebruikt worden. Een oplossing kan zijn om de hele woordgeschiedenis als context te beschouwen en de kansen aan te passen op de volgende manier:

$$P(t_l|H_q) = \frac{1}{p+1} \frac{P(w_q|t_l)P(t_l|H_{q-1})}{\sum_{k=1}^{L} P(w_q|t_k)P(t_k|H_{q-1})} + \frac{p}{p+1} P(t_l|H_{q-1}) \tag{5.2}$$

hierbij is p een parameter die experimenteel bepaald zal moeten worden. Hoe kleiner p, hoe groter de invloed zal zijn van een nieuw woord, terwijl een grote p ervoor

zorgt dat oudere woorden meer invloed hebben op de kansen $P(t_l|H_q)$. Deze parameter p heeft een gelijkaardige functie als de lengte van de woordgeschiedenis bij LSA.

In de tests die volgen zal steeds de invloed van één parameter op de perplexiteit bestudeerd worden. De onderzochte parameters zijn het aantal topics L, de parameter p voor het aanpassen van de kansen $P(t_l|H_q)$ en de interpolatieparameter λ, gebruikt bij de lineaire interpolatie tussen N-grammen en pLSA. Voor de eerste twee tests zal het gecombineerde model IWGM uit vergelijking 4.17 gebruikt worden. Tenslotte zal in de vierde en laatste test onderzocht worden welk combinatiemodel tussen N-grammen en pLSA het best scoort.
Ook hier worden, net als bij LSA, de resultaten van het alleenstaande pLSA-model niet weergegeven wegens niet competitief genoeg.

5.5.1 Aantal topics L

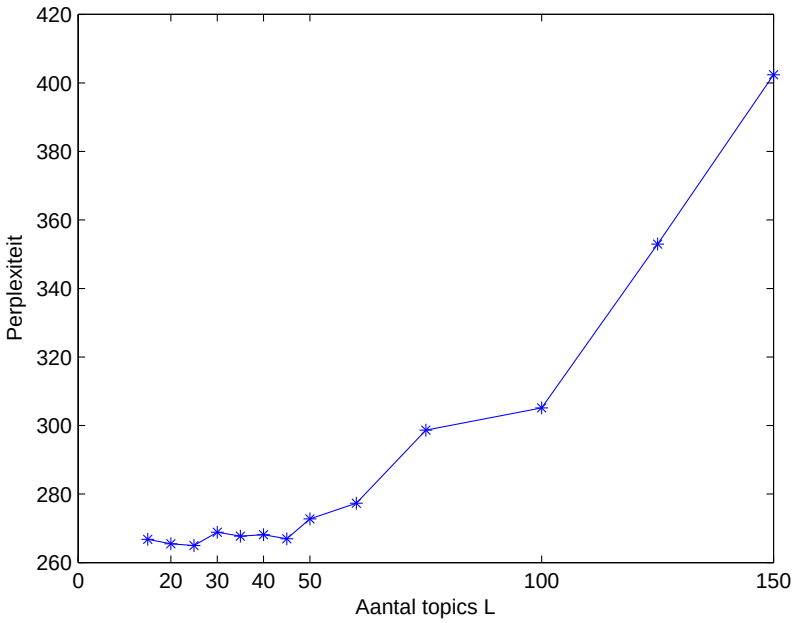
De resultaten van de test voor het aantal topics zijn in figuur 5.7 weergegeven. Na enkele korte tests bleek 10 een goede startwaarde te zijn voor de parameter p, deze waarde is gebruikt in deze test. Het is meteen duidelijk dat een lagere orde dan bij LSA gewenst is. De beste resultaten worden verkregen tussen 15 en 45 topics. De resultaten schommelen meer dan bij LSA, dit komt omdat het NMF-algoritme slechts een lokaal minimum teruggeeft. Het lokaal minimum dat bereikt wordt is afhankelijk van de startoplossing, die random wordt gekozen. Dit verklaart het schommelende gedrag tussen 15 en 45 topics. Het minimum wordt uiteindelijk bereikt voor slechts 25 topics, dit aantal zal in het finale model gebruikt worden. Dit aantal is weinig in vergelijking met waarden die in de literatuur gebruikt worden. In [12] worden 250 topics gebruikt, in [15] 256.

5.5.2 Parameter p

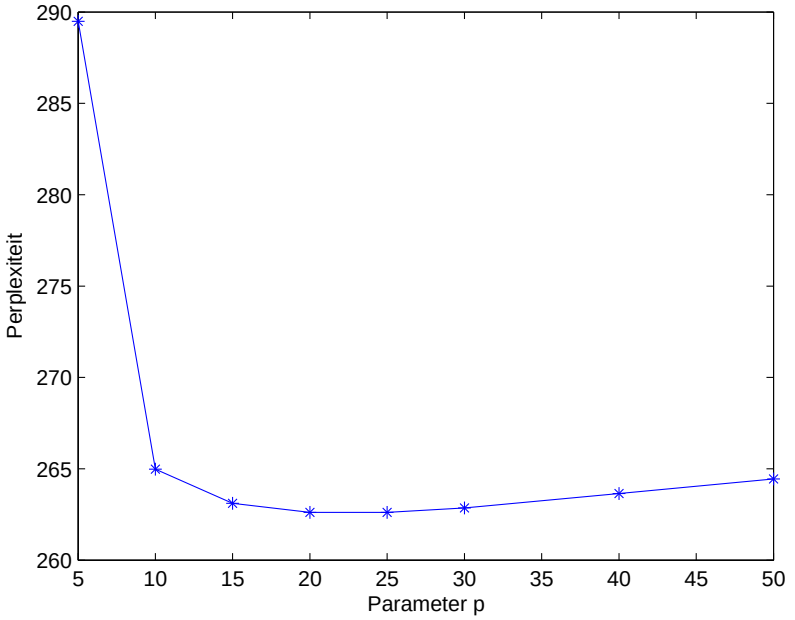
De invloed van de parameter p, die een maat is voor hoe snel de kansen $P(t_l|H_q)$ aangepast worden aan nieuwe woorden, is getest in een pLSA-model bestaande uit 25 topics. In figuur 5.8 is te zien dat het beste resultaat verkregen wordt wanneer $p = 20$. Deze waarde zal in het vervolg steeds gebruikt worden.

5.5.3 Interpolatieparameter λ

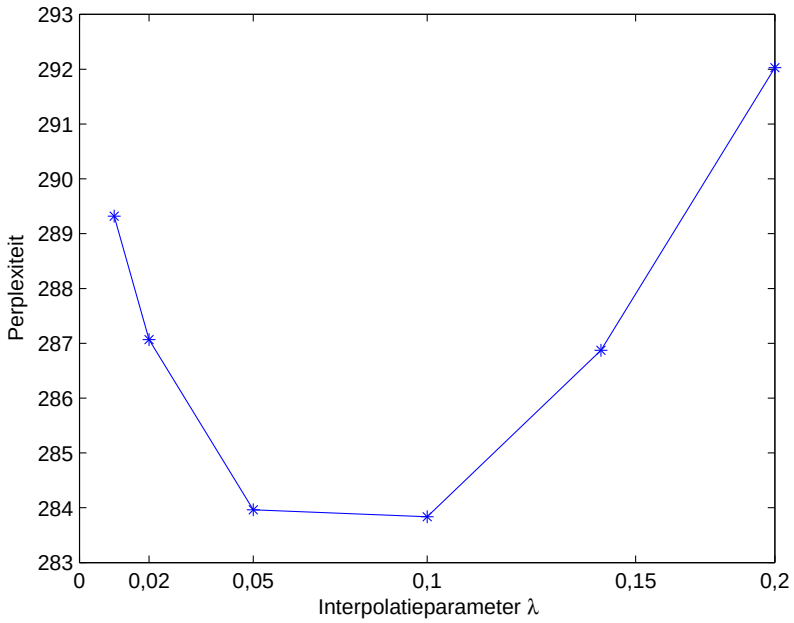
Net als bij LSA moet de beste waarde van de interpolatieparameter λ in het gecombineerde N-gram en pLSA-model gevonden worden. De resultaten zijn weergegeven in figuur 5.9. In het gebruikte pLSA-model zijn 25 topics aanwezig en is $p = 20$. De laagste perplexiteit wordt bereikt voor $\lambda = 0.1$, al is het verschil met $\lambda = 0.05$ verwaarloosbaar. In het vervolg wordt λ op 0.1 gezet.



Figuur 5.7: Invloed van het aantal topics op het pLSA-model.



Figuur 5.8: Invloed van de parameter p bij het pLSA-model.



Figuur 5.9: Invloed van de interpolatieparameter λ op het gecombineerde N-gram en pLSA-model.

5.5.4 Combinatiemodellen met N-grammen

De verschillende combinatiemodellen, beschreven in vergelijkingen 4.14-4.18, zijn getest op de validatieset. De resultaten zijn te zien in tabel 5.4. Hier zijn gelijkaardige conclusies als bij LSA te trekken. Het model met de quasi-Bayesiaanse interpretatie en het SIMMOD-model scoren opnieuw het slechtst, al doen ze het hier relatief gezien een pak beter dan bij LSA. Het beste model is opnieuw IWGM, gevolgd door IWAM en lineaire interpolatie. Het finale pLSA-model zal IWGM als combinatiemodel gebruiken.

Combinatiemodel	**Perplexiteit**
quasi-Bayes	698.11
LI	283.49
IWAM	274.84
IWGM	**262.61**
SIMMOD	451.11

Tabel 5.4: Vergelijking van de combinatiemodellen tussen N-grammen en pLSA.

5.6 Overzicht

In de vorige secties werden via een validatieset voor elk model de parameterwaarden geoptimaliseerd. Voor LSA en pLSA is ook het beste combinatiemodel met N-grammen gezocht, voor cachingmodellen is enkel de lineaire interpolatie met N-grammen onderzocht. Nu al deze tests doorlopen zijn, kunnen de finale modellen met elkaar vergeleken worden. De vergelijking komt tot stand door voor elk model de perplexiteit op de twee onafhankelijke testsets te berekenen. De volgende modellen zijn getest: een 3-gram met gewijzigde Kneser-Ney smoothing en interpolatie, een uniform cachingmodel gecombineerd met een 3-gram, een exponentieel cachingmodel gecombineerd met een 3-gram, een LSA-taalmodel gecombineerd met een 3-gram via IWGM en een pLSA-taalmodel eveneens gecombineerd met een 3-gram via IWGM.

5.6.1 Testset Knack

In tabel 5.5 bevinden zich de resultaten van de testset uit Knack. Naast de perplexiteit is ook de relatieve winst ten opzichte van het basis 3-gram gegeven.
Uniforme cachingmodellen leveren een relatieve winst van ongeveer 8% op tegenover het basis 3-gram. Indien een exponentiële cache gebruikt wordt, is er een bijkomende winst van 1%. Dit is iets minder dan de winst beschreven in [17], [26] en [28], waar het cachingmodel 12 tot 13% beter scoort dan het 3-gram.
Het LSA-model haalt met een relatieve winst van 18% een nog betere score. Dit is meer dan de 10% winst op een 3-gram die in [7] gehaald wordt. Dit resultaat is verrassend, aangezien de datasets niet in documenten onderverdeeld waren. Indien dat wel het geval zou zijn, zit er mogelijk een nog grotere winst in.
De resultaten voor het pLSA-model zijn heel wat minder goed dan die van het LSA-model. De relatieve winst bedraagt 10%, dit is nauwelijks meer dan de winst bij de veel eenvoudigere cachingmodellen.

Gebruikt model	**Perplexiteit**	**Relatieve winst**
3-gram	290.56	
Uniforme cache + 3-gram	267.82	7.83%
Exponentiële cache + 3-gram	264.88	8.84%
LSA + 3-gram (IWGM)	238.04	18.08%
pLSA + 3-gram (IWGM)	261.72	9.93%

Tabel 5.5: Perplexiteitresultaten op de testset uit Knack.

5.6.2 Testset NRC Handelsblad

De resultaten van de verschillende taalmodellen op de testset uit NRC Handelsblad zijn weergegeven in tabel 5.6. De perplexiteit ligt hier voor elk model iets hoger dan bij de testset uit Knack. Dit resultaat is niet verrassend, aangezien de modellen getraind zijn op Belgische kranten, Knack een Belgisch tijdschrift is en NRC Handelsblad een Nederlandse krant is. In Knack zullen bijvoorbeeld veel artikels voorkomen

over de Belgische politiek, waarover veel data aanwezig is in de trainingsset van De Standaard. NRC Handelsblad zal dan weer meer schrijven over de Nederlandse politiek, waarover minder data aanwezig is in de trainingsset van De Standaard. Toch is het verschil niet al te groot, voor het basis 3-gram ligt de perplexiteit minder dan 10% hoger bij de Nederlandse krant dan bij Knack.

Cachingmodellen scoren hier relatief gezien iets beter dan bij de vorige testset. Een uniform cachingmodel haalt een winst van net geen 11%, terwijl een exponentieel cachingmodel ruim 12% winst haalt. Ook het LSA-model scoort beter dan bij Knack. Het haalt net geen 20% winst tegenover N-grammen.
De resultaten voor het pLSA-model zijn hier eveneens teleurstellend. Het pLSA-model haalt 11% winst, dit is evenveel als het uniform cachingmodel en slechter dan het exponentieel cachingmodel. Het zwakke resultaat van het pLSA-taalmodel is waarschijnlijk veroorzaakt door de niet-optimale modelparameters. Het NMF-algoritme vindt immers slechts een lokaal optimum. Ook is het algoritme wegens computationele redenen per parameterset slechts één keer doorlopen. Het is beter om in de plaats het algoritme meerdere keren vanuit verschillende startoplossingen uit te voeren en dan het beste model te selecteren. Andere suggesties om het pLSA-model te verbeteren worden gedaan in [16], waarin een aangepaste versie van het EM-algoritme gebruikt wordt om overfitting tegen te gaan, en in [11], waar de LSA-oplossing gebruikt wordt om de startoplossing van het EM-algoritme te initialiseren. Aangezien pLSA ook gebruik maakt van de frequentiematrix W, kan ook hier het model verbeterd worden indien de datasets in afzonderlijke documenten verdeeld zijn.

Gebruikt model	**Perplexiteit**	**Relatieve winst**
3-gram	313.72	
Uniforme cache + 3-gram	279.28	10.98%
Exponentiële cache + 3-gram	275.37	12.22%
LSA + 3-gram (IWGM)	252.43	19.54%
pLSA + 3-gram (IWGM)	279.30	10.97%

Tabel 5.6: Perplexiteitresultaten op de testset uit NRC Handelsblad.

Het is ook interessant om de verwerkingstijden tussen de verschillende modellen te vergelijken. In tabel 5.7 staan zowel de tijden voor het aanmaken van het model als de tijden voor de berekening van de perplexiteit van de testset uit NRC Handelsblad. Voor de modellen die gecombineerd worden met N-grammen is de tijd nodig voor het berekenen van de N-gram kansen niet ingerekend. De berekeningen zijn uitgevoerd op een computer met 4 cores en een geheugen van 16 GB. Bij cachingmodellen dient er geen model te worden aangemaakt waardoor deze verwerkingstijd nul is. Het valt op dat de verwerkingstijden voor het LSA- en pLSA-model hoog zijn in vergelijking met N-grammen en cachingmodellen. Vooral de tijd nodig voor de berekening van de perplexiteit ligt zeer hoog. Dit komt omdat het gebruikte combinatiemodel tussen

N-grammen en (p)LSA IWGM is, en dit model de N-gram kansen voor alle woorden nodig heeft. Het berekenen van deze kansen neemt het meeste tijd in beslag.

Model	**Tijd trainen model**	**Tijd berekenen pp**
3-gram	6 minuten	14 seconden
Uniforme cache + 3-gram	/	11 seconden
Exponentiële cache + 3-gram	/	21 seconden
LSA + 3-gram (IWGM)	3 uur	23 uur
pLSA + 3-gram (IWGM)	2 uur	23 uur

Tabel 5.7: Verwerkingstijden van de verschillende modellen.

Hoofdstuk 6

Algemeen besluit

Het doel van deze masterproef was om een geschikt taalmodel te vinden voor het Nederlands. Het taalmodel zou dan kunnen ingewerkt worden in een automatische spraakherkenner. Hierbij werd op zoek gegaan naar een zuiver statistisch taalmodel, dat geen rekening houdt met de aanwezige grammatica in een taal. De verkregen taalmodellen werden getest op Nederlandstalige tekstdata. De vergelijking tussen taalmodellen onderling gebeurde in termen van de behaalde perplexiteit op een testset. Een zo laag mogelijke waarde voor de perplexiteit was gewenst.

6.1 Taalmodellen

De eenvoudige en meest gebruikte taalmodellen, N-grammen, kwamen het eerst aan bod. Deze taalmodellen tellen hoeveel keer een bepaald woord in een context is voorgekomen en geven op basis hiervan de kans op dit woord terug. Op deze manier modelleren ze heel goed de lokale afhankelijkheden aanwezig in taal. Een nadeel van N-grammen is de beperkte afhankelijkheid van $N-1$ woorden; van zodra een afhankelijkheid zich uitstrekt over meer woorden, kunnen N-grammen dit niet opvangen. Een ander nadeel is de sparsiteit van de data. Er is enorm veel data nodig om uit de frequentietellingen tot betrouwbare woordschattingen te komen. Dikwijls wordt er bij gebrek aan data teruggevallen op lagere orde tellingen, wat niet ideaal is. Een laatste, niet onbelangrijk, nadeel van N-grammen is dat door de lokale afhankelijkheden vooral syntactische fenomenen worden gedetecteerd, terwijl semantiek niet aan bod komt.
Cachingmodellen zijn een andere eenvoudige vorm van taalmodellen. Deze modellen tellen hoeveel keer een bepaald woord in de cache voorkomt; dit zijn de meest recente herkende woorden. Op basis hiervan geeft het model een kans op het volgende woord terug. Cachingmodellen kunnen gecombineerd worden met N-grammen. Het gecombineerde model kan in termen van perplexiteit een winst halen van ongeveer 10% op het standaard N-gram.

Latent semantische analyse is een techniek om verborgen betekenisovereenkomsten tussen woorden te ontdekken. Het start hierbij van een frequentiematrix waar in

de rijen alle woorden uit het vocabularium $\mathcal{V}$ staan en in de kolommen de documenten uit de trainingsset. Na een weging wordt een afgebroken SWO van deze matrix berekend. Deze rang k benadering reduceert de ruis aanwezig in de oorspronkelijke matrix en laat toe om woorden en documenten met elkaar te vergelijken. De cosinus tussen woorden en/of documenten is een maat voor de betekenisverwantschap tussen woorden/documenten. Het LSA-taalmodel berekent voor elk nieuw woord de cosinus tussen dit woord en de documentgeschiedenis. Op basis hiervan geeft het een kans voor dit woord terug. Een voordeel van LSA-taalmodellen is dat het afhankelijkheidsgebied, de documentgeschiedenis, bijna altijd groter zal zijn dan bij N-grammen. Op deze manier kan het rekening houden met de semantische inhoud van woorden. Een belangrijk nadeel van LSA-taalmodellen is de rekencomplexiteit. Het aanmaken van een afgebroken SWO neemt veel tijd in beslag. Een ander probleem van LSA is dat de rang k benadering van de frequentiematrix de Frobeniusnorm tussen de benadering en de oorspronkelijke matrix minimaliseert. Hierbij gaat het ervan uit dat de ruis in de frequentiematrix Gaussisch verdeeld is, wat dikwijls niet het geval is. LSA slaagt er ook niet in om polysemie op te vangen. LSA zal de verschillende betekenissen van eenzelfde woord als hetzelfde beschouwen. Een laatste en belangrijk nadeel is dat LSA-taalmodellen de woordvolgorde links laten liggen. Daarom worden ze dikwijls gecombineerd met N-grammen. Dit kan op verschillende manieren gebeuren. Het beste gecombineerde model kan tot 20% beter scoren dan een basis N-gram.

Probabilistische LSA probeert enkele tekortkomingen van LSA op te vangen. Het topic model zegt dat documenten in topics zijn onderverdeeld en dat elk woord een bepaalde kans heeft om binnen een topic op te treden. Omdat hetzelfde woord in verschillende topics kan voorkomen, kan polysemie opgevangen worden. De documentverdelingen in topics en de topicverdelingen in woorden zijn de onbekende parameters van het model. De likelihood van het model wordt gemaximaliseerd met het EM-algoritme. Het nadeel van dit algoritme is dat het slechts een lokaal optimum als oplossing teruggeeft. In tegenstelling tot bij LSA is er bij de maximalisatie echter geen veronderstelling meer van Gaussisch verdeelde ruis in de data nodig. Een andere methode om tot een oplossing te komen en die minder geheugen vraagt is de niet-negatieve matrixfactorisatie. Net als het LSA-model is de rekencomplexiteit hoog ten opzichte van het N-gram taalmodel.
Om de kans op het volgende woord terug te geven, berekent het pLSA-taalmodel eerst de verdeling van de documentgeschiedenis in topics. Vervolgens kan het met behulp van deze verdeling en met de topicverdelingen in woorden de gezochte kans vinden. Net als het LSA-taalmodel kan het pLSA-taalmodel op verschillende manieren gecombineerd worden met N-grammen. De resultaten van het gecombineerde model vallen echter tegen. Een relatieve winst op N-grammen van ongeveer 10% wordt gehaald, wat ongeveer even goed is als de veel eenvoudigere cachingmodellen. Dit tegenvallende resultaat is mogelijk te wijten aan het vastlopen van de modeloplossing in een lokaal minimum. Er bestaan echter wel methodes om deze lokale oplossing te verbeteren.

6.2 Verder onderzoek

Cachingmodellen werden steeds gecombineerd met N-grammen via lineaire interpolatie. Bij zowel LSA als pLSA scoorde echter een ander combinatiemodel beter dan de eenvoudige lineaire interpolatie. Het is mogelijk dat dit ook voor cachingmodellen het geval is. Een mogelijkheid is om bijvoorbeeld de interpolatieparameter λ afhankelijk van het woord te maken.

In deze masterproef werden N-grammen telkens gecombineerd met slechts één ander model. Het loont de moeite om eens na te denken over de integratie van meer dan twee taalmodellen binnen één groot taalmodel. Zo is het misschien mogelijk om LSA te combineren met zowel N-grammen als cachingmodellen. Op die manier wordt er zowel rekening gehouden met de woordvolgorde, semantiek als met woordherhalingen. Ook de mogelijkheid tot combinatie tussen pLSA, N-grammen en cachingmodellen kan onderzocht worden.

De resultaten van het pLSA-taalmodel vielen tegen. Er werd verwacht dat de resultaten minstens in de lijn van die van LSA zouden liggen. Dikwijls is de lokale oplossing van het pLSA-taalmodel niet goed. In [11] en [16] wordt een suggestie gedaan om een beter pLSA-model te verkrijgen. Het kan de moeite lonen om deze technieken eens uit te testen op het pLSA-model voor het Nederlands.

Een andere suggestie tot onderzoek is het verder uitwerken van het idee beschreven in sectie 3.7. Daar werd een LSA-model aangemaakt door in de kolommen van de frequentiematrix een korte context te plaatsen of zowel een korte context als hele documenten. Op die manier konden op basis van de cosinus tussen twee woorden de woorden gezocht worden die zowel syntactisch als semantisch gelijkaardig aan een woord zijn. Een strategie dient nog ontwikkeld te worden om met de informatie van het LSA-model de kansen uit het N-gram taalmodel aan te passen.

Bijlage A

Poster

Uitbreiding van N-gram taalmodellen met semantische informatie

KATHOLIEKE UNIVERSITEIT LEUVEN

FACULTEIT INGENIEURSWETENSCHAPPEN

Master Wiskundige ingenieurstechnieken

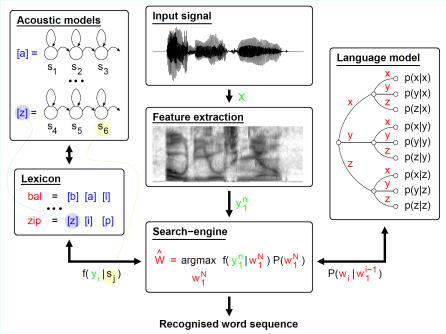
Masterproef *Bruno De Laet*

Promotor *Prof. Patrick Wambacq*

Academiejaar 2011-2012

1. Situering

- **Spraakherkenning**



2. Eenvoudige taalmodellen

- **Standaard N-gram taalmodel**:

$$P(w_q \mid w_1^{q-1}) \approx P(w_q \mid w_{q-N+1}^{q-1})$$

- **Caching modellen**
  - Kijken hoeveel keer een woord in laatste K woorden (=cache) is voorgekomen

- **Doel**: op zoek naar taalmodellen die rekening houden met de betekenis van woorden (semantiek)

3. Geavanceerde taalmodellen

- **Latent semantische analyse** (LSA)
  - Idee: vinden van verborgen semantische verbanden tussen woorden. Dit kan via afgebroken SWO van frequentiematrix.
  - Afstand binnen gereduceerde ruimte is maat voor betekenisovereenkomst

- **Probabilistische latent semantische analyse** (pLSA)
  - Idee: documenten zijn een mengeling van onderwerpen, en een onderwerp is een kansverdeling van woorden

$$P(w_j \mid d_i) = \sum_{k=1}^{K} P(w_j \mid z_k) P(z_k \mid d_i)$$

4. Vergelijking taalmodellen

- **N-gram**
  - \+ Woordvolgorde, eenvoudig
  - \- Context slechts N-1 woorden, sparsiteit data, geen semantiek
- **LSA - pLSA**
  - \+ Semantiek, veel woorden in context
  - \- CPU-intensief, geen woordvolgorde

➡ Voordelen N-gram + (p)LSA in 1 model combineren

5. Evaluatie taalmodel

- **Entropie**: $H(X) = -\sum_{x \in \chi} p(x) \log_2(x)$

-> Toegepast op taalmodellering:

$$H(L) = \lim_{n \to \infty} -\frac{1}{n} \log p(w_1 w_2 ... w_n)$$

- **Perplexiteit**: 2^H

6. Experiment

- **Trainingsset**: 65 miljoen woorden uit De Standaard
- **Validatieset**: 100k woorden uit De Morgen
- **Testset:** 50k woorden uit Knack

- **Resultaten:**

Taalmodel	Perplexiteit testset
3-gram	290,56
Caching + 3-gram	264,88
LSA + 3-gram	238,04
pLSA + 3-gram	261,72

Bijlage B

Wetenschappelijk artikel

Enriching N-gram language models with semantic information

Bruno De Laet, Patrick Wambacq, Kris Demuynck and Joris Pelemans
KU Leuven - Dept. ESAT
Kasteelpark Arenberg 10, box 2441, B-3001 Leuven, Belgium

***Abstract*—The aim of this article is to find a language model for Dutch that can be integrated into an automatic speech recognizer. First some simple language models, N-grams and cache-based models, are studied. Then some techniques are investigated that lead to more advanced language models. These techniques include latent semantic analysis (LSA) and probabilistic latent semantic analysis (pLSA). Finally, the different language models are compared by calculating their perplexity on different test sets.**

I. Introduction

There is an increasing need for natural language processing. An example is the dictation of a text to a computer or the analysis of a spoken text by a computer. Automatic speech recognition (ASR) is the translation of spoken text into written text by computers. An extented overview of ASR is given in [1]. The speech signal is first converted by a front-end parametrisation in acoustic vectors $y_1 y_2..y_n = y_1^n$. The task of an ASR system is to find the most probable word sequence $w_1 w_2..w_n = w_1^n$, given the acoustic vectors y_1^n:

$$\hat{W} = \arg\max_{w_1^n} P(w_1^n|y_1^n) \tag{1}$$

With Bayes' rule this can be written as:

$$\hat{W} = \arg\max_{w_1^n} \frac{P(y_1^n|w_1^n)P(w_1^n)}{P(y_1^n)} \tag{2}$$

Since the term $P(y_1^n)$ is independent of w_1^n, the optimal solution is found as the maximum of the product of $P(y_1^n|w_1^n)$ and $P(w_1^n)$. The first term corresponds to the acoustic model and the second one to the language model. The task of the decoder is to combine the results returned by the acoustic and language model in order to find the optimal solution of equation 2.

The aim of this article is to find an appropriate language model for Dutch that can be inserted into an ASR system. Most of the language models are already known in literature, but only few of them are tested on Dutch data.

The article is organized as follows. Section II describes some simple language models, N-grams and cache-based models. Sections III and IV describe two techniques that both lead to an advanced language model: latent semantic analysis (LSA) and probabilistic latent semantic analysis (pLSA). All the models are evaluated in section V. Finally a conclusion follows in section VI.

II. Simple language modeling

A. N-grams

The most widely used language models are N-grams [2]. A N-gram approximates the probability of the qth word w_q by looking at the $N-1$ previous words w_{q-N+1}^{q-1}:

$$P(w_q|w_1^{q-1}) \approx P(w_q|w_{q-N+1}^{q-1}) \tag{3}$$

N-gram models are trained by counting all the N-grams in the training set. The probability of occurrence of a specific N-gram is calculated as the total number of occurrences $C(w_{q-N+1}^{q})$ of that specific N-gram, divided by the total number of occurrences $C(w_{q-N+1}^{q-1})$ of the $(N-1)$-gram where the last word w_q is left out:

$$P(w_q|w_{q-N+1}^{q-1}) = \frac{C(w_{q-N+1}^{q})}{C(w_{q-N+1}^{q-1})} \tag{4}$$

A major problem of N-gram language models is that they are trained on a limited data set. Therefore not all possible N-grams are present in this data set. From equation 4, an unseen N-gram would receive a probability of zero. This is not desired, since every N-gram needs to have a probability greater than zero. There are two major techniques to solve this problem:

1) Smoothing: Smoothing methods will allocate some of the probability mass of high frequent N-grams to unseen or low frequent N-grams. The easiest method is additive smoothing. This method adds a small constant value, usually one, to the count of all possible N-grams. However this method seems not to work well [3]. In [4] an overview of more advanced smoothing techniques is given. These techniques include Witten-Bell discounting, Good-Turing discounting, absolute discounting and modified Kneser-Ney smoothing.

2) Backoff and interpolation: Another method to solve the problem of zero counts is the use of lower order counts. If a specific N-gram is not in the data set, but the $(N-1)$-gram without the first word of the N-gram is, the probability of the N-gram can be approximated by the count of the $(N-1)$-gram. Now there are two possibilities. The first possibility only uses the lower order counts if the count of the standard order is zero. This is called backoff. Another possibility is to use both the standard order count and lower order counts and combine them by means of a linear interpolation. This is called interpolation.

N-gram language models are easily understandable and fast

to construct. They model local dependencies in language very well. However, when a dependency is longer than N words, N-grams are unable to handle this. Another disadvantage of N-grams is the sparsity of the data. A lot of N-grams don't occur in the training corpus and their probability is estimated by lower order counts by means of smoothing or backoff/interpolation. This is definitely not optimal. A last, but not unimportant, disadvantage of N-grams is that they don't take into account the meaning of words. More advanced techniques are needed to model this.

B. Cache-based models

Usually it happens that when a specific word occurs in a text, it will reoccur soon in the same text. Cache-based models are based on this principle. They keep a cache of size K with the K most recently occurred words $w_{q-K}, ..., w_{q-1}$. For each new word w_q, they count the total number of times $C_{\text{cache}}(w_q)$ that this word occurs in the cache. The probability of occurrence of this new word then equals this number divided by the size K of the cache [5]:

$$P_{cache}(w_q|w_{q-K}^{q-1}) = \frac{C_{\text{cache}}(w_q)}{K} \tag{5}$$

This is a uniform cache-based model. Every word in the cache contributes as much to the cache probability of the next word. Normally, more recent words have a greater impact on the next word and therefore they should contribute more to the cache probability than older words. In an exponential cache-based model, the contribution of a word decreases exponentially when going further back in time [5]:

$$P_{cache}(w_q|w_1^{q-1}) = \beta \sum_{j=1}^{q-1} I(w_q = w_j) e^{-\alpha(q-j)} \tag{6}$$

where $I(x)$ is an indicator function that returns 1 if x is true and 0 otherwise, α is the decay rate and β is a normalization factor.

Standalone cache-based models are very sparse, since they only take into account the last K words. Consequently they should be combined with N-grams. The most used combination method is linear interpolation [5]:

$$\begin{aligned} P(w_q|w_1^{q-1}) &= (1-\lambda) P_{n-gram}(w_q|w_{q-N+1}^{q-1}) \\ &\quad + \lambda P_{cache}(w_q|w_1^{q-1}) \end{aligned} \tag{7}$$

with $0 \leq \lambda \leq 1$. The interpolation parameter λ is determined from a validation set.

III. Latent semantic analysis

N-gram language models only look at the last $(N-1)$ words to predict the next word. Moreover they are unable to take into account the meaning of words. Latent semantic analysis (LSA) is a technique that can handle these two issues. LSA is based on the fact that words that occur in similar contexts tend to have similar meanings [6]. LSA seeks to find hidden similarities between words.

A. Model construction

The first step is the construction of a frequency matrix W. This assumes that the training corpus $\mathcal{T}$ is divided into separate documents. Each row in W represents a unique word of the vocabulary $\mathcal{V}$, with $|\mathcal{V}| = M$. The columns of W represent the N documents of $\mathcal{T}$. Each element c_{ij} of W equals the number of times that word w_i occurs in document d_j. Consequently W will be a large and sparse matrix.

The next step is the weighting of the elements in W. The idea is to give more weight to surprising events and less weight to expected events. The weighting consists of two steps: a local weighting $L(i,j)$ applied on each element in W and a global weighting $G(i)$ applied on each row in W. The weighted count W_{ij} will then be:

$$W_{ij} = G_i L_{ij} \tag{8}$$

As local weighting usually the following is used [7] [8]:

$$L_{ij} = \log_2(1 + c_{ij}) \tag{9}$$

The global weighting G_i that will be used is [9]:

$$G_i = 1 - E_i \tag{10}$$

with

$$E_i = -\frac{1}{\log_2(N)} \sum_{j=1}^{N} \frac{c_{ij}}{t_i} \log_2 \frac{c_{ij}}{t_i} \tag{11}$$

where t_i is the total number of times that w_i occurs in $\mathcal{T}$.

The following step is the computation of a truncated singular value decomposition (SVD) of W:

$$W \approx USV^T \tag{12}$$

where U is a $(M \times k)$ matrix, S a $(k \times k)$ matrix and V a $(N \times k)$ matrix. Only the k largest singular values are preserved, the others are set to zero. This leads to a rank k best approximation of W. This rank reduction suppresses the noise in W and the remaining matrices are not sparse anymore. In the reduced space $\mathcal{S}$, each left singular vector u_i is a representation of a word w_i, while each right singular vector v_j is a representation of document d_j.

In $\mathcal{S}$, a distance between words and/or documents can be defined. Words with a similar meaning will lie close to each other in this space. The same applies for documents. The most popular distance measure is the cosine between two vectors [6]. The cosine between two vectors $\mathbf{x}$ and $\mathbf{y}$ is given by:

$$\cos(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x}}{\|\mathbf{x}\|} \cdot \frac{\mathbf{y}}{\|\mathbf{y}\|} \tag{13}$$

It is a measure of the angle between two vectors and is independent of their length. The greater the cosine, the closer two vectors will be.

B. LSA language model

Now a LSA language model can be created. The probability calculation in a LSA model is:

$$P(w_q|H_{q-1}) = P(w_q|H_{q-1}, \mathcal{S}) \tag{14}$$

where H_q is the admissible context. The dependency on $\mathcal{S}$ refers to the dimension k of $\mathcal{S}$. Usually the context consists of the current document up to word w_{q-1}. This context needs to have a representation in the reduced space $\mathcal{S}$. First consider the document $\tilde{d}_p$, this is the current document up to word w_p. Since $\tilde{d}_p$ is a new document and not part of W, it is called a pseudodocument. From equation 12, a pseudodocument $\tilde{d}_p$ can be written as:

$$\tilde{d}_p = US\tilde{v}_p^T \tag{15}$$

Without loss of generality, the approximation symbol from equation 12 is dropped. The representation of the pseudodocument $\tilde{d}_p$ in the space $\mathcal{S}$ can be written as:

$$\tilde{v}_p = \tilde{d}_p^T US^{-1} \tag{16}$$

This is the pseudodocument representation. The pseudodocument $\tilde{d}_{q-1}$ can be used as the context H_q:

$$P(w_q|H_{q-1}, \mathcal{S}) = P(w_q|\tilde{d}_{q-1}) \tag{17}$$

This probability can be calculated from the distance between the representations of w_q and $\tilde{d}_{q-1}$ in $\mathcal{S}$, i.e. u_q and $\tilde{v}_{q-1}$.

The (i,j)-th element of the frequency matrix W is a measure for the extent to which word w_i and document d_j co-occur in the corpus $\mathcal{T}$. From equation 12, this element equals the dot product between $u_i S^{1/2}$ and $v_j S^{1/2}$. This is the same as the cosine between the vectors $u_i S^{1/2}$ and $v_j S^{1/2}$. Thus a metric K to consider for the closeness of a word w_i and a document d_j is:

$$K(u_q, \tilde{v}_{q-1}) = \cos(u_q S^{1/2}, \tilde{v}_{q-1} S^{1/2}) \tag{18}$$

$$= \frac{u_q S \tilde{v}_{q-1}^T}{\left\|u_q S^{1/2}\right\| \left\|\tilde{v}_{q-1} S^{1/2}\right\|} \tag{19}$$

Now a probability for the word w_q can be derived:

$$P(w_q|\tilde{d}_{q-1}) = \frac{[K(u_q, \tilde{v}_{q-1}) - K_{\min}(u, \tilde{v}_{q-1})]^\gamma}{\sum_{w_i \in \mathcal{V}} [K(u_i, \tilde{v}_{q-1}) - K_{\min}(u, \tilde{v}_{q-1})]^\gamma} \tag{20}$$

The cosine between the word w_q and the document history $\tilde{d}_{q-1}$ in the space $\mathcal{V}$ is calculated. This is divided by the sum of the cosines between all the words from the vocabulary $\mathcal{V}$ and the document history $\tilde{d}_{q-1}$. As the cosine can be negative and in order to get a probability, the smallest cosine found between all the words and the pseudodocument is subtracted from each cosine. Usually this is raised to a power $\gamma > 1$ to increase the dynamic range of the probabilities [10].

C. Discussion

One large advantage of the LSA language model is that, in contrast with N-grams, it predicts the next word based on its meaning. The probability of the next word depends on the closeness of this word to the document history. Also the context, i.e. the document history, is usually longer than the last $N-1$ words for N-grams. Another quality of the LSA model is that it can capture synonyms well. In the reduced space synonyms lie close to each other and consequently they are nearly the same for the LSA model. One example are the sentences *I take a picture* and *I take a photograph*. While these two sentences are completely different for N-grams, they are almost equal for the LSA language model.

However, there are some disadvantages too. The word order is completely ignored, which can lead to the prediction of syntactically wrong words. Also the computational complexity is higher than for N-grams. A truncated SVD must be computed. Moreover once the model is found, to assign a probability to the next word, the cosine must be calculated between all the words from the vocabulary and the document history. This is more work than some table lookups needed for N-grams. Another disadvantage of LSA language models has to do with the rank k approximation of W. The Frobenius norm between the approximation and the old matrix W is minimized, but this assumes that the noise in the frequency matrix is Gaussian, which is usually not the case. A last disadvantage is that LSA language models can't capture polysemy. For example they don't make a difference between a river bank and a bank as a financial institution.

D. Combination with N-grams

A standalone LSA language model is a poor predictor of words. It doesn't take the word order into account and consequently local dependencies are ignored. The idea is to combine the locally oriented N-grams with the more globally oriented LSA model. The following combinations are possible:

1) quasi-Bayesian interpretation [9]: :

$$P(w_q|H_{q-1}) \propto \frac{P_{N\text{-gram}} P_{\text{LSA}}}{P_{\text{uni}}} \tag{21}$$

where P_{uni} is the unigram probability $P(w_q)$.

2) Linear interpolation (LI) [11]: :

$$P(w_q|H_{q-1}) = (1-\lambda) P_{N\text{-gram}} + \lambda P_{\text{LSA}} \tag{22}$$

3) Information weighted arithmetic mean (IWAM) [10]: :

$$P(w_q|H_{q-1}) \propto (1-\lambda_q) P_{N\text{-gram}} + \lambda_q P_{\text{LSA}} \tag{23}$$

where $\lambda_q = \frac{1}{2}[1 - E_i]$. λ_q is called the LSA confidence and it indicates how reliable the LSA probability is.

4) Information weighted geometric mean (IWGM) [10]: :

$$P(w_q|H_{q-1}) \propto P_{N\text{-gram}}^{(1-\lambda_q)} + P_{\text{LSA}}^{\lambda_q} \tag{24}$$

5) Similarity modulated N-gram (SIMMOD) [11]: :

$$P(w_q|H_{q-1}) \propto P_{N\text{-gram}} P_{\text{LSA}} \tag{25}$$

In each of the cases, except for linear interpolation, one has to normalize by the sum over the whole vocabulary.

IV. Probabilistic latent semantic analysis

The LSA language model can't capture polysemy. Also LSA assumes Gaussian noise in the frequency matrix by minimizing a Frobenius norm. These two problems are not present in the case of probabilistic latent semantic analysis (pLSA). There are many similarities between LSA and pLSA, but a major difference is that pLSA works from a statistical point of view.

A. Topic model

The topic model is the starting point of pLSA. It is based on the idea that documents are a mixture of topics, and each topic is a probability distribution over words [12]. It is a generative model for documents. To create a new document, a distribution over topics is chosen. To generate a new word in the document, first a topic is chosen and then a word is generated in that topic. Suppose there are L topics. The probability of observing a word in document d_j is $P(d_j)$, $P(w_i|t_l)$ is the topic specific probability distribution over the words and $P(t_l|d_j)$ represents the document specific distribution over topics. With these definitions, a joint probability model for the co-occurrences of word/documents can be defined:

$$P(w_i, d_j) = P(d_j)P(w_i|d_j) \tag{26}$$

where

$$P(w_i|d_j) = \sum_{l=1}^{L} P(w_i|t_l)P(t_l|d_j) \tag{27}$$

The parameters $P(d_j)$, $P(w_i|t_l)$ and $P(t_l|d_j)$ are the unknowns of the pLSA model. They must be calculated from the frequency matrix W. As training criterion the log-likelihood $\mathcal{L}$ can be used, this is the log-probability of the data W under the model θ:

$$\mathcal{L}(\theta; W) = \sum_{i=1}^{M}\sum_{j=1}^{N} n(w_i, d_j) \log P(w_i, d_j) \tag{28}$$

$$= \sum_{i=1}^{M}\sum_{j=1}^{N} n(w_i, d_j) \log P(d_j) + \sum_{i=1}^{M}\sum_{j=1}^{N} n(w_i, d_j) \log \left[\sum_{l=1}^{L} P(w_i|t_l)P(t_l|d_j)\right] \tag{29}$$

where $n(w_i, d_j)$ is the weighted number of times that word w_i occurs in document d_j. The first term leads to a solution $P(d_j) \propto n(d_j)$, with $n(d_j)$ the total number of words in document d_j. A solution for the second term can be found using the Expectation-Maximization (EM) algorithm [13]. The E-step goes as follows:

$$P(t_l|w_i, d_j) = \frac{P(w_i|t_l)P(t_l|d_j)}{\sum_{k=1}^{L} P(w_i|t_k)P(t_k|d_j)} \tag{30}$$

while the M-step is:

$$P(w_i|t_l) = \frac{\sum_{j=1}^{N} n(w_i, d_j)P(t_l|w_i, d_j)}{\sum_{m=1}^{M}\sum_{j=1}^{N} n(w_m, d_j)P(t_l|w_m, d_j)} \tag{31}$$

$$P(t_l|d_j) = \frac{\sum_{i=1}^{M} n(w_i, d_j)P(t_l|w_i, d_j)}{\sum_{k=1}^{L}\sum_{i=1}^{M} n(w_i, d_j)P(t_k|w_i, d_j)} \tag{32}$$

The E- and M-step are alternated until a stop condition is reached.

B. pLSA language model

The pLSA language model can now be constructed. The parameters $P(t_l|d_j)$ are discarded, as they don't apply to new documents. However, the topic distributions $P(t_l|H_i)$ of these documents are not known. They can be estimated as follows [14]:

$$P(t_l|H_i) = \frac{1}{i+1}\frac{P(w_i|t_l)P(t_l|H_{i-1})}{\sum_{k=1}^{L} P(w_i|t_k)P(t_k|H_{i-1})} + \frac{i}{i+1}P(t_l|H_{i-1}) \tag{33}$$

$$P(t_l|H_1) = P(t_l) = \frac{\sum_{i=1}^{M}\sum_{j=1}^{N} n(w_i, d_j)P(t_l|d_j)}{\sum_{i=1}^{M}\sum_{j=1}^{N} n(w_i, d_j)} \tag{34}$$

The probability of occurrence of a word w_q can then be calculated as:

$$P(w_q|H_{q-1}) = \sum_{l=1}^{L} P(w_q|t_l)P(t_l|H_{q-1}) \tag{35}$$

Standalone pLSA language models are just as LSA language models unsufficient to predict words. Consequently they are usually combined with N-grams as well. The same combinations as between LSA and N-grams in equations 21-25 can be used.

C. Discussion

There are a lot of similarities between LSA and pLSA. In pLSA, the subdivision of a document in topics can be seen as a dimensionality reduction, similar to the truncated SVD in LSA. pLSA can also be seen as a matrix factorization. The $(M \times N)$ frequency matrix W is separated into a $(M \times L)$ word-topic matrix T and a $(L \times N)$ topic-document matrix H.

A crucial difference between LSA and pLSA lies in the objective function. LSA minimizes a Frobenius norm, implicitly assuming Gaussian noise in the frequency matrix. On the other hand, the goal function in pLSA is a likelihood function, in order to get a model with the strongest predicting power. Another difference between both models lies in the ability to capture polysemy. By using topics, pLSA can make a difference between the different meanings of the same word. A word can have a reasonable occurrence probability in more than one topic. Therefore different meanings of the same word can occur in different topics. One disadvantage of pLSA compared to LSA is that the EM-algorithm only succeeds in finding a local optimum, while the truncated SVD can be computed exactly.

Compared to N-grams, the computational complexity is much higher. The EM-algorithm requires multiple iterations, and if the local solution is bad a new solution must be computed. To assign a probability to the next word, the topic distributions $P(t_l|H_i)$ need to be computed first, which requires more work than the table lookups for N-grams.

The word order is completely ignored in the pLSA language model, but this issue is solved by combining the pLSA model with N-grams.

V. Experiments

The different language models can be compared. One way to evaluate a language model is by measuring its perplexity on a data set. Perplexity is defined as:

$$PP = 10^{-\frac{1}{Q}\sum_{q=1}^{Q}\log_{10}P(w_q|H_{q-1})} \tag{36}$$

where Q is the size of the data set. Perplexity is a measure for how well a language model can predict the next word. If the perplexity of a language model equals z, guessing the next word with use of this language model is as difficult as choosing between z equiprobable possibilities.

First the unknown parameters of each model are tuned by minimizing perplexity on a validation set. Then the perplexity for each model on two different test sets is computed.

A. Data

The tests are performed on Dutch language data. All the data sets come from the Mediargus data from the period 1999-2004. The training set consists of 65 million words from De Standaard, a Belgian newspaper. With this data the N-gram, LSA and pLSA model can be trained. About 100 000 words from De Morgen, another Belgian newspaper, form the validation set. This set is used to tune the unknown parameters of each model. The final models are compared by their perplexity on the two test sets. The first test set consists of 50 000 words from Knack, a Belgian magazine, and the second test set consists of 50 000 words from NRC Handelsblad, a Dutch newspaper.

An important design choice that has to be made, is the size of the vocabulary $\mathcal{V}$. A small vocabulary will lead to a relatively high value of the out-of-vocabulary (OOV) rate, this is the percentage of words from the test set that are not included in $\mathcal{V}$. In the following experiments the 105 000 most occurring words in the training set form the vocabulary $\mathcal{V}$. This way the OOV rate is smaller than 5% for both test sets.

B. Validation results

1) N-grams: A 3-gram language model is tested with different smoothing techniques and backoff/interpolation. The best results are obtained for a 3-gram combined with modified Kneser-Ney smoothing and interpolation.

2) Cache-based models: For a uniform cache-based model, the best results are obtained using a cache size K of 400 and an interpolation parameter $\lambda = 0.1$. When using an exponential cache-based model, the optimal cache size K is 1000, the decay rate $\alpha = 0.006$ and the interpolation parameter $\lambda = 0.1$.

3) LSA: An important parameter in the LSA language model is the dimension size k of the reduced matrix. The perplexity values on the validation set for different values of k are shown in figure 1. The perplexity values are found for a combined LSA with N-gram model, with IWGM used as combination. There is a minimum for $k = 150$, although the perplexity differences with other values of k are small. The optimal value for the exponent γ is 11, while the best value for the interpolation parameter $\lambda = 0.05$.

All the combination models, described in subsection III-D, are tested on the validation set and the IWGM model obtained the best result, see table I.

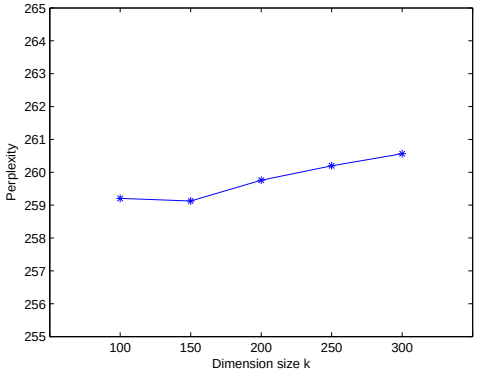


Fig. 1. Influence of the dimension size k in the LSA model.

Combination model	Perplexity
quasi-Bayes	4138.69
LI	272.56
IWAM	300.22
IWGM	239.45
SIMMOD	982.05

TABLE I
Comparison of the different combination models between N-grams and LSA.

4) pLSA: The optimal number of topics needs to be found. The perplexity for different numbers of topics is calculated using a combined N-gram and pLSA model, with IWGM used as combination. From figure 2 it is clear that a low number of topics is preferred. The lowest perplexity is reached for 25 topics.

The different combination models are tested for pLSA as well. For the simple linear interpolation, the best value for $\lambda = 0.1$. In table II the perplexity results are given for the different combinations. The results are comparable with these from LSA, IWGM is the best model again. This is not surprising, since the IWGM method is the only one that takes a non-linear combination of the N-gram and (p)LSA probability and that uses the LSA confidence metric.

Combination model	Perplexity
quasi-Bayes	698.11
LI	283.49
IWAM	274.84
IWGM	262.61
SIMMOD	451.11

TABLE II
Comparison of the different combination models between N-grams and PLSA.

C. Test results

1) Knack: The different models are now ready to be compared. In table III, the perplexity results on the Knack test

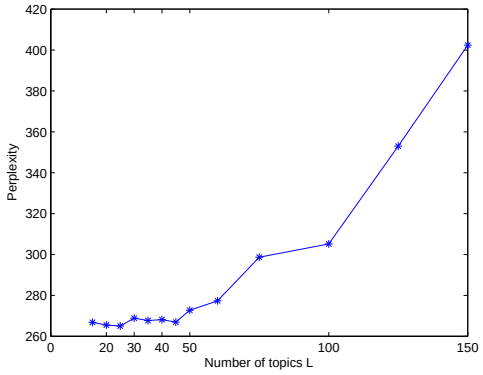


Fig. 2. Influence of the number of topics on the pLSA model.

Used model	Perplexity	Relative gain
3-gram	290.56	
Uniform cache + 3-gram	267.82	7.83%
Exponential cache + 3-gram	264.88	8.84%
LSA + 3-gram (IWGM)	238.04	18.08%
pLSA + 3-gram (IWGM)	261.72	9.93%

TABLE III
PERPLEXITY RESULTS ON THE TEST SET FROM KNACK.

set are given for the baseline 3-gram model, a uniform cache-based model and an exponential cache-based model combined with a 3-gram, and a LSA and pLSA model combined with a 3-gram according IWGM. The relative gain on the baseline 3-gram model is given too. It is clear that the LSA model is the best, it scores 18% better than the baseline 3-gram. The results of the cache-based models and pLSA are comparable. The relative gain is 8-10%. The exponential cache-based model is a little better than the uniform cache-based model.

2) NRC Handelsblad: The results for the NRC Handelsblad test set are given in table IV. The LSA model is again the best with a relative gain of almost 20%. The cache-based models and the pLSA model are performing less with a gain of 10-12%. The relatively poor result of the pLSA model can probably be explained by the poor local solution of the EM-algorithm. In [13] and in [15] some suggestions are made to improve the solution of the pLSA model.

Used model	Perplexity	Relative gain
3-gram	313.72	
Uniform cache + 3-gram	279.28	10.98%
Exponential cache + 3-gram	275.37	12.22%
LSA + 3-gram (IWGM)	252.43	19.54%
pLSA + 3-gram (IWGM)	279.30	10.97%

TABLE IV
PERPLEXITY RESULTS ON THE TEST SET FROM NRC HANDELSBLAD.

VI. CONCLUSION

Different language models are compared with each other. The simple models, N-grams and cache-based models, are easy to use and fast to construct. However, they lack the ability to take into account the meaning of words. More advanced techniques are needed for this purpose. LSA and pLSA are two examples of such techniques and from both techniques a language model can be derived. These language models perform the best in combination with N-grams. The best results in terms of perplexity on a test set are obtained for the LSA model with a 20% gain on the baseline 3-gram. The results of the cache-based models and pLSA model are comparable, with a gain of about 10% in perplexity on the 3-gram.

Suggestions for further work include the investigation of other combinations of cache-based models and N-grams. In this article only a linear combination between both has been used. The possibility of integration of N-grams, cache-based models and (p)LSA in one large language model can be investigated as well. Further, the suggestions done in [13] and [15] to improve the local pLSA solution can be implemented on the pLSA model for Dutch data.

REFERENCES

[1] S. Young, "Large vocabulary continuous speech recognition a review," *IEEE Signal Processing Magazine*, vol. 13, no. 5, pp. 1–4, 1996.
[2] D. Jurafsky and J. Martin, *Speech and Language Processing: An Introduction to Natural Language Processing, Speech Recognition, and Computational Linguistics*, 2nd ed. Prentice-Hall, 2009.
[3] W. A. Gale and K. W. Church, "What's wrong with adding one," in *Corpus-Based Research into Language. Rodolpi*, 1994.
[4] S. F. Chen and J. Goodman, "An empirical study of smoothing techniques for language modeling," Tech. Rep., 1996.
[5] P. Clarkson and A. J. Robinson, "Language model adaptation using mixtures and an exponentially decaying cache," in *In Proceedings of ICASSP-97*, 1997, pp. 799–802.
[6] P. D. Turney and P. Pantel, "From frequency to meaning: Vector space models of semantics," *CoRR*, vol. abs/1003.1141, 2010.
[7] J. Bellegarda, "Exploiting both local and global constraints for multi-span statistical language modeling," *Proceedings of ICASSP'98*, vol. 2, pp. 677–680, 1998.
[8] S. T. Dumais, "Improving the retrieval of information from external sources," *Behavior Research Methods, Instruments, & Computers*, vol. 23, no. 2, pp. 229–236, 1991. [Online]. Available: http://psychonomic.org/search/view.cgi?id=5145
[9] J. Bellegarda, "A multispan language modeling framework for large vocabulary speech recognition," *IEEE Transactions on Speech and Audio Processing*, vol. 6, no. 5, pp. 456–467, Sep. 1998.
[10] N. Coccaro and D. Jurafsky, "Towards better integration of semantic predictors in statistical language modeling," in *In Proceedings of ICSLP-98*, 1998, pp. 2403–2406.
[11] M. Pucher and Y. Huang, "Combination of latent semantic analysis based language models for meeting recognition," in *Computational Intelligence*, 2006, pp. 349–354.
[12] M. Steyvers and T. Griffiths, "Probabilistic topic models," in *Latent Semantic Analysis: A Road to Meaning.*, T. Landauer, D. Mcnamara, S. Dennis, and W. Kintsch, Eds. Laurence Erlbaum, 2006. [Online]. Available: http://cocosci.berkeley.edu/tom/papers/SteyversGriffiths.pdf
[13] T. Hofmann, "Unsupervised learning by probabilistic latent semantic analysis," in *Machine Learning*, 2001, p. 2001.
[14] D. Gildea and T. Hofmann, "Topic-based language models using EM," in *In proceedings of Eurospeech*, 1999, pp. 2167–2170.
[15] A. Farahat and F. Chen, "Improving probabilistic latent semantic analysis with principal component analysis," in *EACL*, 2006.

Bibliografie

[1] P. H. Algoet and T. M. Cover. Asymptotic optimality and asymptotic equipartition properties of log-optimum investment. *The Annals of Probability*, 16(2):876–898, Apr. 1988.

[2] J. Bellegarda. Exploiting both local and global constraints for multi-span statistical language modeling. *Proceedings of ICASSP'98*, 2:677–680, 1998.

[3] J. Bellegarda. A multispan language modeling framework for large vocabulary speech recognition. *IEEE Transactions on Speech and Audio Processing*, 6(5):456–467, Sept. 1998.

[4] J. Bullinaria and J. Levy. Extracting semantic representations from word co-occurrence statistics: A computational study. *Behavior Research Methods*, (3):510, 2007.

[5] S. F. Chen and J. Goodman. An empirical study of smoothing techniques for language modeling. Technical report, 1996.

[6] P. Clarkson and A. J. Robinson. Language model adaptation using mixtures and an exponentially decaying cache. In *In Proceedings of ICASSP-97*, pages 799–802, 1997.

[7] N. Coccaro. *Latent semantic analysis as a tool to improve automatic speech recognition performance*. PhD thesis, Boulder, CO, USA, 2005. AAI3190360.

[8] N. Coccaro and D. Jurafsky. Towards better integration of semantic predictors in statistical language modeling. In *In Proceedings of ICSLP-98*, pages 2403–2406, 1998.

[9] C. H. Q. Ding, T. Li, and W. Peng. On the equivalence between non-negative matrix factorization and probabilistic latent semantic indexing. *Computational Statistics & Data Analysis*, 52(8):3913–3927, 2008.

[10] S. T. Dumais. Improving the retrieval of information from external sources. *Behavior Research Methods, Instruments, & Computers*, 23(2):229–236, 1991.

[11] A. Farahat and F. Chen. Improving probabilistic latent semantic analysis with principal component analysis. In *EACL*, 2006.

[12] M. Federico. Language model adaptation through topic decomposition and mdi estimation. volume 1, pages 773–776, 2002.

[13] W. A. Gale and K. W. Church. What's wrong with adding one. In *Corpus-Based Research into Language. Rodolpi*, 1994.

[14] E. Gaussier and C. Goutte. Relation between PLSA and NMF and implications. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '05, pages 601–602, New York, NY, USA, 2005. ACM.

[15] D. Gildea and T. Hofmann. Topic-based language models using EM. In *In proceedings of Eurospeech*, pages 2167–2170, 1999.

[16] T. Hofmann. Unsupervised learning by probabilistic latent semantic analysis. In *Machine Learning*, page 2001, 2001.

[17] F. Jelinek, B. Mérialdo, S. Roukos, and M. Strauss. A dynamic language model for speech recognition. In *HLT*, 1991.

[18] D. Jurafsky and J. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Speech Recognition, and Computational Linguistics.* Prentice-Hall, 2nd edition, 2009.

[19] R. Kuhn and R. D. Mori. A cache-based natural language model for speech recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12:570–583, 1990.

[20] T. K. Landauer, P. W. Foltz, and D. Laham. An introduction to latent semantic analysis. *Discourse Processes*, (25):259–284, 1998.

[21] D. D. Lee and H. S. Seung. Algorithms for non-negative matrix factorization. In *NIPS*, pages 556–562, 2000.

[22] R. Ordelman, A. V. Hessen, and F. D. Jong. Lexicon optimization for Dutch speech recognition in spoken document retrieval. In *in Proc. European Conference on Speech Communication and Technology*, pages 1085–1088, 2001.

[23] M. Pucher and Y. Huang. Combination of latent semantic analysis based language models for meeting recognition. In *Computational Intelligence*, pages 349–354, 2006.

[24] M. Steyvers and T. Griffiths. Probabilistic topic models. In T. Landauer, D. Mcnamara, S. Dennis, and W. Kintsch, editors, *Latent Semantic Analysis: A Road to Meaning.* Laurence Erlbaum, 2006.

[25] A. Stolcke. SRILM - an extensible language modeling toolkit. pages 901–904, 2002.

[26] C. Tillmann and H. Ney. Statistical language modeling and word triggers. In *Proc. Int. Workshop Speech and Computer*, pages 22–27, St. Petersburg, Oct. 1996.

[27] P. D. Turney and P. Pantel. From frequency to meaning: Vector space models of semantics. *CoRR*, abs/1003.1141, 2010.

[28] A. Vaiciunas and G. Raskinis. Cache-based statistical language models of English and highly inflected Lithuanian, 2005.

[29] S. Young. Large vocabulary continuous speech recognition a review. *IEEE Signal Processing Magazine*, 13(5):1–4, 1996.

Fiche masterproef

Student: Bruno De Laet

Titel: Uitbreiding van N-gram taalmodellen met semantische informatie

Engelse titel: Enriching N-gram language models with semantic information

UDC: 51-7

Korte inhoud:

De verwerking van natuurlijke taal wordt steeds belangrijker. Een voorbeeld is het dicteren van een tekst aan een computer. Een automatische spraakherkenner zal trachten om gesproken taal in tekst om te zetten. Dit proces verloopt in verschillende stappen. Een onderdeel van een automatische spraakherkenner is het taalmodel. Het taalmodel geeft de a priori kans van een woordsequentie terug. Het doel van deze masterproef is om een geschikt taalmodel te vinden, en dit voor het Nederlands. Dit taalmodel kan dan geïntegreerd worden in een automatische spraakherkenner.

Gestart is met de meest gebruikte taalmodellen, N-grammen. Deze modellen voorspellen het volgende woord op basis van de $N-1$ vorige woorden. Het aanmaken van het model gebeurt op basis van frequentietellingen op grote hoeveelheid tekstdata. Smoothing en backoff/interpolatie zijn technieken om met de spaarsheid van N-grammen om te gaan.
Cachingmodellen zijn gebaseerd op het feit dat pas voorgekomen woorden een grote kans hebben om weldra nog eens voor te komen. Omdat deze modellen nog spaarser zijn dan N-grammen, worden ze dikwijls met N-grammen gecombineerd.

In een volgend hoofdstuk is latent semantische analyse (LSA) besproken. LSA gaat ervan uit dat gelijkaardige woorden in gelijkaardige documenten voorkomen en gaat op zoek naar verborgen betekenisovereenkomsten tussen woorden. Het LSA-taalmodel wijst een kans toe aan het volgende woord op basis van de afstand tussen dit woord en de documentgeschiedenis. Omdat LSA-taalmodellen lokale afhankelijkheden binnen taal negeren, worden ze dikwijls gecombineerd met N-grammen.

Daarna komt probabilistische latent semantische analyse (pLSA) aan bod. Deze techniek werkt enkele tekortkomingen van LSA weg. Het werkt in tegenstelling tot LSA vanuit een statistisch oogpunt. Het topic model gaat ervan uit dat documenten een mengeling zijn van topics, die zelf een mengeling van woorden zijn. Op basis van de aanwezige topics in de documentgeschiedenis zal het pLSA-taalmodel de kans op het volgende woord teruggeven. Ook hier wordt het taalmodel meestal gecombineerd met N-grammen.

In de experimenten zijn de verschillende taalmodellen onderling vergeleken. In een eerste stadium worden alle modelparameters geoptimaliseerd door het minimaliseren van de perplexiteit op een validatieset. Nadien kan van elk finaal model de perplexiteit berekend worden op twee onafhankelijke testsets. Elk taalmodel wordt vergeleken ten opzichte van het basis N-gram taalmodel. Het LSA-taalmodel haalt duidelijk de beste resultaten. De resultaten van het cachingmodel en het pLSA-taalmodel zijn vergelijkbaar.

Thesis voorgedragen tot het behalen van de graad van Master in de ingenieurswetenschappen: wiskundige ingenieurstechnieken

Promotor: Prof. dr. ir. Patrick Wambacq

Assessoren: Prof. dr. ir. Johan Suykens
Prof. dr. ir. Marc Van Barel

Begeleiders: Dr. ir. Kris Demuynck
Joris Pelemans