Deelverslag 2 Groepsproject Collectieve Intelligentie

Groepsleden:

- Rabie Afqir (**11606843**)
- Björn Out (**12567930**)
- Frank Tamer (**12248738**)
- Gino Pennasilico (**12393819**)

2.0 Update taakverdeling

De taakverdeling is zeer beperkt aangepast. Rabie gaat nu de aanpak van het cold start probleem verzorgen. Deze behandelen we in de verslagen als een losse aanpak.

2.1 Algoritmes

SVM

A) Beschrijf het algoritme. Het liefst met pseudocode.

1. Bepaal voor welke gebruiker aanbevelingen moeten worden gedaan
2. Haal alle reviews van de gebruiker op, en vul deze aan met alle *features* van de gereviewde bedrijven.
3. Bepaal aan de hand van een *threshold* of de reviews positief of negatief zijn en kies de grootste klasse (positief of negatief).
4. Train een *Single-Class SVM* op basis van de *features* van de bedrijven in de grootste klasse.
5. Predict aan de hand van de getrainde *SVM* voor ieder nog niet gereviewd bedrijf in alle steden waarin de gebruiker ooit een bedrijf heeft gereviewd of het bedrijf kan worden aangeraden aan deze gebruiker.
6. Kies een willekeurig bedrijf uit de potentieel aan te raden bedrijven en voeg deze toe aan de definitief aan te raden bedrijven.
7. Herhaal tot het gewenst aantal aanbevelingen is bereikt óf er geen potentieel aan te raden bedrijven meer over zijn:
    a. Bepaal voor ieder resterend potentieel aan te raden bedrijf de gemiddelde *Jaccard similarity* met de momenteel definitief aan te raden bedrijven.
    b. Kies het bedrijf met de laagste gemiddelde *Jaccard similarity* en voeg deze toe aan de definitief aan te raden bedrijven.
8. Einde.

B) Zijn er randgevallen waar je rekening mee moet houden? In welke stappen van de bovenstaande pseudocode is dat? Hoe los je die gevallen op?

Het zou kunnen dat er in stap 5 te weinig potentieel aan te raden bedrijven worden gevonden, waardoor er in stap 7 niet genoeg bedrijven zijn om definitief aan te raden. Dit zal voornamelijk gebeuren wanneer de gebruiker nog te weinig reviews heeft geplaatst om de *SVM* zinvolle voorspellingen te laten doen. In dit geval kunnen de lege plekken opgevuld worden met bedrijven gevonden middels onze cold-start aanpak (verderop in dit document is deze aanpak uitgebreid omschreven).

Het is ook aannemelijk dat de SVM in stap 5 veel op elkaar lijkende bedrijven zal 'nomineren' om aan te raden. Dit wordt opgelost door de uiteindelijke selectie van de definitief aan te raden bedrijven te baseren op de *Jaccard similarity* tussen een potentieel aan te raden bedrijf en de reeds definitief aan te raden bedrijven (in stap 7).

C) Wat zijn de parameters van je algoritme? Hoe beïnvloeden deze parameters het algoritme?

- De *threshold* van de classificatie wel/niet aanraden: wanneer deze *threshold* naar beneden wordt bijgesteld zal het algoritme een bedrijf sneller aanbevelen, waardoor de uiteindelijke relevantie van de aangeraden bedrijven zal afnemen.
- De *kernel* van de *SVM*: wanneer een andere *kernel* dan *'linear'* wordt gebruikt is het aannemelijk dat de kwaliteit van de aanbevelingen achteruit zal gaan, omdat het algoritme door een hogere complexiteit zal *overfitten* op de training data.
- De gamma-waarde van de SVM: de gamma-waarde bepaalt de invloed die één trainingsvoorbeeld kan hebben op de uiteindelijke *SVM* (*inverted*). Hoe lager deze waarde wordt gezet, hoe meer flexibiliteit de *optimizer* heeft om een optimaal model te bouwen. Wel loopt hiermee de kans op *overfitting* op, dus er zal een goede balans moeten worden gevonden in deze waarde.
- De 'nu'-waarde van de SVM: de 'nu'-waarde van een *one-class SVM* is een *regularization parameter*. Hoe lager deze waarde, hoe complexer het algoritme en hoe meer kans op *overfitting*. Hoe hoger deze waarde, hoe beter het algoritme te generaliseren zal zijn naar nieuwe data.

D) Waren er problemen met het implementeren?

- Bepaalde categorieën in de kolom 'attributes' zijn niet geschikt voor het gebruiken met *Jaccard similarity*. Omdat dit voornamelijk categorieën waren die niet veel bedrijven überhaupt hadden, hebben we ervoor gekozen deze weg te laten.
- Het algoritme was in eerste instantie door intensieve data-bewerkingen voor iedere gebruiker te traag om op grote schaal te evalueren. Dit is opgelost door te evalueren op basis van slechts enkele steden in plaats van de gehele dataset.
- In eerste instantie gebruikten we een reguliere SVM. Sommige gebruikers hadden echter alleen positieve of alleen negatieve reviews, waardoor voor sommige gebruikers geen SVM kon worden getraind. Om dit op te lossen hebben we Single-Class SVM's gebruikt op de grootste klasse.

Item-based Collaborative Filtering

A) Beschrijf het algoritme. Het liefst met pseudocode.

1. Kies een stad
2. Laad de data in
3. Maak utility matrix met de reviews
4. Maak een similarity matrix met cosine similarity
5. Selecteer de genormaliseerde neighbourhood van een gebruiker
6. Bereken hiermee de predicted value voor een bepaalde business en gebruiker

B) Zijn er randgevallen waar je rekening mee moet houden? In welke stappen van de bovenstaande pseudocode is dat? Hoe los je die gevallen op?

Er wordt met deze methode gefocust op enkel features binnen één bepaalde stad, omdat er anders simpelweg veel te veel data ingeladen moet worden. Dit gebeurt dan ook bij de eerste stap van het algoritme. Ook is het voor een gebruiker vaak niet heel relevant om aanbevelingen uit andere steden te krijgen wanneer men hier niet naar op zoek is. De nauwe focus op een enkele stad is op te lossen door een aantal steden rondom de desbetreffende stad ook in te laden, waardoor er een grote bereik van aanbevelingen gerealiseerd wordt en tegelijk efficiënt wordt omgegaan met rekenkracht.

C) Wat zijn de parameters van je algoritme? Hoe beïnvloeden deze parameters het algoritme?

Een paar parameters van het algoritme zijn bijvoorbeeld een bepaalde dataset, waarin de data van één (of in de toekomst meerdere) stad zit. Het algoritme werkt op basis van deze data en de resultaten van het algoritme zijn hier dan ook erg afhankelijk. Mogelijke parameters zijn ook de user en een business zijn unieke id. Deze beïnvloeden het algoritme doordat op basis van deze waarden de neighbourhood wordt bepaald. Vooralsnog is het aantal teruggegeven items in de neighborhood ook nog van belang.

D) Waren er problemen met het implementeren?

In de dataset zijn er veel gebruikers die simpelweg te weinig bedrijven een beoordeling hebben gegeven. Hierdoor is het moeilijker om aan de hand van dit algoritme nuttige aanbevelingen te doen. Dit is opgelost door het cold start approach te gebruiken wanneer je een gebruiker een aanbeveling wil geven die niet genoeg beoordelingen gegeven heeft.

Cold start Approach

A) Beschrijf het algoritme. Het liefst met pseudocode.

1. Sorteer de bedrijven op hoeveel reviews ze hebben.
2. Neem de N-aantal meest beoordeelde bedrijven.
3. Calculeer de lower bound confidence interval
4. Sorteer de lower bound confidence interval van de beoordelingen van de bedrijven op aflopende wijze
5. Loop door de bedrijven heen en recommend de bedrijven waarvan categorieën niet vaker dan K aanbevolen zijn.

6. Geef de X aantal bedrijven met de hoogste lower bound confidence interval van de beoordelingen van de bedrijven.

B) Zijn er randgevallen waar je rekening mee moet houden? In welke stappen van de bovenstaande pseudocode is dat? Hoe los je die gevallen op?
Ja, er zijn bedrijven die nog wel in de database staan maar die niet meer open zijn. Wij waren dat eerst wel vergeten. Natuurlijk checken we dat nu wel. Daarnaast kan het ook zo zijn dat in het aantal bedrijven dat tot dan toe beschikbaar is, niet genoeg diversiteit is. Het is namelijk zo dat de bedrijven die minimaal 1800 reviews hebben, niet divers genoeg zijn als je 10 aanbevelingen wilt waarin een categorie maar één keer voor mag komen. Dit is opgelost door voorrang te geven aan de diversiteit en niet aan de kwantiteit. Dat betekent dus dat we 9 diverse aanbevelingen doen in plaats van 10 minder gevarieerde aanbevelingen in bovenstaand voorbeeld.

C) Wat zijn de parameters van je algoritme? Hoe beïnvloeden deze parameters het algoritme?
Het Cold Start algoritme heeft drie parameters: “NumberOfPreds” oftewel het gewenst aantal aanbevelingen, “RepetitionsOfCategories” oftewel hoe vaak een categorie voor mag komen in de aanbevelingen en “MinimalAmountOfReviews” oftewel hoeveel aanbevelingen een bedrijf moet hebben om in aanmerking te komen voor een aanbeveling.

“NumberOfPreds” zorgt voor het aantal aanbevelingen teruggeeft. Dit is variabel omdat een website of een app verschillende aantallen aanbevelingen kunnen willen. “RepetitionsOfCategories” zorgt voor de variatie in de aanbevelingen door aan te geven hoe vaak een categorie in de set van aanbevelingen voor mag komen. Als laatste is daar nog “MinimalAmountOfReviews”. Door een minimum aantal reviews te vragen wordt niet alleen de prestatie van het algoritme beter, maar ook de kwaliteit. Als dit niet geïmplementeerd zou zijn, zou een bedrijf met vijf 5-star ratings beter kunnen zijn dan een bedrijf met 3000 beoordelingen waarvan 95% 5 sterren was.

D) Waren er problemen met het implementeren?
Godzijdank niet.

2.2 Baseline

SVM

Methode	Accuracy	Precision	Recall
SVM	0.5816	0.6747	0.7307
Baseline	0.4961	0.6674	0.5014

Het belangrijkste doel van dit algoritme is het aanraden van relevante bedrijven aan gebruikers. Dit moet zich uitdrukken in een hoge *precision*. We zien dat de SVM hierin niet

beter presteert dan de baseline. Wel is de *accuracy* hoger en de *recall* is veel hoger. Dit betekent dat het algoritme wel minder vaak ten onrechte concludeert dat een bedrijf niet interessant is, terwijl dat wel zo is.

Gemiddelde Jaccard SVM: 0.0000
Gemiddelde *baseline* Jaccard: 0.0596
Op basis van 50 willekeurig gekozen gebruikers, 5 aanbevelingen per gebruiker.

Te zien is dat onze aanpak leidt tot aanbevelingen met geen enkele overlap tussen de verschillende aangeraden bedrijven. Dit betekent dat er helemaal geen overlap in categorieën bestaat tussen deze verschillende aanbevelingen. Bij willekeurige selectie ontstaat er meer overlap. Het deel van het algoritme dat de selectie doet (stappen 6 en 7 in de pseudocode) heeft geen parameters. In theorie zal deze dus altijd hetzelfde moeten presteren. Experimentatie met verschillende *input data* bevestigt dit vermoeden.

Alle waarden zijn bepaald met parameters threshold = 3.5, gamma = 0.000001, kernel = 'rbf' en nu = 0.2.

Item-based collaborative filtering

Methode	MSE
Collaborative Filtering item based	3.26
Random predictions (Baseline)	4.36
Item means predictions (Baseline)	1.98

Gebruikte K-waarde parameter = 5 , Gebruikte stad = 'Brooklyn'

Bij de evaluatie van ons item-based filteringsysteem hebben wij een Mean Squared Error ondervonden van 3.26. Dit is beduidend lager dan wanneer er willekeurige voorspellingen plaats zouden vinden, waarbij er sprake was van een Mean Squared Error van 4.36. Wanneer het gemiddelde wordt genomen van alle reviews die een business heeft gekregen, dan ontstaat er een Mean Squared Error van 1.69. De lage score die in de eerstgenoemde methode voorkwam duidt op een relatief succesvolle manier van voorspellen. Dit valt te verklaren dat er veel gebruikers zijn die maar enkele beoordelingen hebben. Voor deze pool van gebruikers is het voor collaborative filtering lastig om nuttige voorspelling te doen. In de praktijk is het meest effectieve om een hybride algoritme te gebruiken waarbij het cold start approach met item-based collaborative filtering gecombineerd wordt.

Cold start Approach

Zoals op de grafieken te zien is, is de Cold Start Approach veel beter dan de random aanbevelingen. Op de X-as staat de nummering zoals die per bedrijf in het dataframe staat.

	Business ID	Predicted Rating	RMSE	MAE	MAPE
1	Xg5qEQiB-7L6kGJ5F4K3bQ	4.750960	0.660154	0.401922	13.430810
2	mDR12Hafvr84ctpsV6YLag	4.605711	0.787607	0.574419	18.715428
3	mz9ltimeAly2c2qf5ctljw	4.328782	1.047926	0.840471	30.831568
4	t-o_Sraneime4DDhWrQRBA	3.932575	1.389513	1.129438	54.882167
5	WYw3Uf56DT5lwpaLNnCH5Q	3.764400	1.270705	1.028700	46.979958

```
Text(1, 1.1, "RMSE, MAE and MAPE's of the predictions (Jaccard Similarity = 0.00365):")
```

RMSE, MAE and MAPE's of the predictions (Jaccard Similarity = 0.00365):