Technische Universiteit Delft

Online anonieme markten

Geo-attributie op productgroep

F.E.G. Miedema

TU Delft

Online anonieme markten

Geo-attributie op productgroep

door

F.E.G. Miedema

ter partiële verkrijging van de graad van Bachelor of Science
aan de Technische Universiteit Delft,

Studentnummer:	4141369	
Projectduur:	april 2018 – juli 2018	
Supervisor:	Dr. ing. A. J. Klievink	TU Delft
Dagelijkse begeleider:	Msc. R. S. van Wegberg	TU Delft
Externe begeleider:	Msc. P. van Lierop	FIOD

Op dit verslag is geheimhouding van toepassing tot en met 30 augustus 2018.

Een elektronische versie van deze scriptie zal beschikbaar worden gemaakt op
`http://repository.tudelft.nl/`.

TU Delft

Voorwoord

Dit rapport is geschreven in het kader van de tweede fase van het vak 'Bachelor-eindproject' van de bacheloropleiding Technische Bestuurskunde aan de Technische Universiteit Delft. Het rapport is geschreven naar aanleiding van het issue-paper dat ik voor de eerste fase van dit project geschreven heb over het bestrijden van financiële, economische en fiscale fraude met een cybercomponent.

Het onderzoek betreft het ontwerp van een methode om de betrokkenheid van een land bij de handel op online anonieme markten in kaart te brengen op productniveau. Deze methode is ontworpen aan de hand van het proces voor Data Mining en maakt voor de analyse gebruik van machine learning technieken en methodes voor content-based geo-location estimation ontworpen voor social media platformen zoals Twitter. De toepasbaarheid en het resultaat van de methode is in de praktijk getest door de toepassing van de methode op de casus van de FIOD.

Nagenoeg alle voorgenoemde zaken waren nieuw voor mij toen ik enkele maanden geleden aan dit onderzoek begon. Ik kijk met veel plezier terug op deze periode en wil graag alle mensen bedanken die mij hierbij bijgestaan, ondersteund of geïnspireerd hebben. Dit rapport was niet tot stand gekomen zonder het enthousiasme, de kundige begeleiding en de nodige begrenzing van Rolf van Wegberg en Bram Klievink. Veel dank ook aan Tineke Ruijgh-van der Ploeg, die gezorgd heeft voor mentale steun, maar met name de nodige aanmoediging gedurende het project. Mijn collega's bij FACT en Pieter van Lierop in het bijzonder wil ik bedanken voor hun expertise en inbreng van de criminologische en strafrechtelijke aspecten, als ook voor de dagelijkse gezelligheid op de afdeling tijdens het schrijven van dit onderzoek.

Ik wens de lezer veel plezier toe met het lezen van het onderzoeksverslag. Bij vragen over de inhoud, conclusies of het onderzoek, ben ik bereikbaar via onderstaande contactgegevens.

F.E.G. Miedema
Delft, juli 2018

f.e.g.miedema@tudelft.nl
06-21846447

Inhoudsopgave

1 Inleiding

1.1. Aanleiding

Wordt het darkweb de digitale tegenhanger van een Braziliaanse *favela*[1]? Waarbij er een verhulde, separate maatschappij en economie ontstaat waar politie, belastingdienst en andere gezaghebbende organen geen grip op krijgen? Wanneer men kijkt naar één van de groeiende fenomenen op het gebied van cybercriminaliteit, namelijk de online anonieme markten op het darkweb, lijkt dit wel het geval. Op deze online anonieme markten wordt een breed scala aan criminaliteit aangeboden en gekocht, zoals bijvoorbeeld drugs, wapens, gegevens en verschillende financiële diensten. Sinds de komst van Silk Road 1 in 2011, zijn de online anonieme markten op het darkweb gegroeid, zowel qua volume als qua omzet aan transacties [37]. De conservatieve schatting is dat de dagelijkse omzet gegenereerd door online anonieme markten in 2015 al rond de $500.000-$600.000 bedroeg en dat deze omzet sindsdien, samen met de populariteit van dit soort markten, verder is toegenomen.

1.1.1. Darkweb

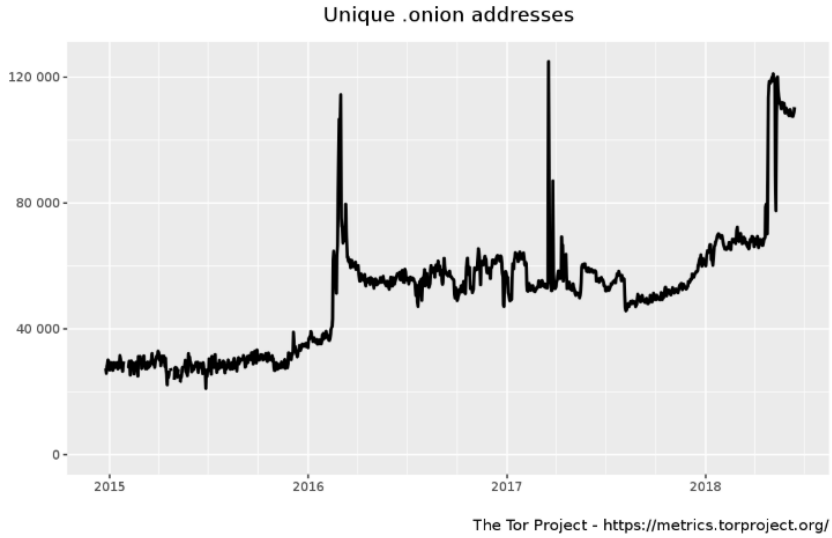
Het darkweb, de omgeving waarbinnen vele online anonieme markten zijn opgezet, bestaat uit meerdere netwerken. Deze netwerken zijn onderdeel van het internet, maar zijn specifiek ontworpen voor het faciliteren van een anoniem, gedecentraliseerd en versleuteld netwerk voor de gebruikers [20]. Anders dan het 'gewone internet', zijn de netwerken van het darkweb alleen beschikbaar wanneer men beschikt over speciale software, specifieke configuratie of autorisatie. Bekende darkweb netwerken (ook wel 'darknets' genoemd) zijn bijvoorbeeld Tor, Freenet of I2P [10]. Momenteel is het Tor netwerk de bekendste en meest gebruikte, al is het meten van de omvang van of het gebruik van een darknet erg methodologisch uitdagend [30]. Op het Tor netwerk kunnen zogeheten 'hidden services' gehost worden, waar 'online anonieme markten' één van de mogelijke types hidden services zijn. Andere voorbeelden van types hidden services zijn webshops, blogs, fora, community's of webpagina's.

Hoewel er legitieme redenen zijn voor het gebruikmaken van een anoniem en versleuteld netwerk, zoals bijvoorbeeld het ondersteunen van journalisten in politiek repressieve regimes, of klokkenluiders van grote schandalen, is de schatting van onderzoekers dat ongeveer de helft van de hidden services van Tor gerelateerd zijn aan een vorm van criminaliteit [30]. Het Tor-project zelf onderzoekt niet waarvoor de hidden services gebruikt worden maar geeft alleen inzage in het aantal unieke .onion adressen. In juni 2018 zijn dat er rond de 100.000 (zie Figuur 1: Unieke .onion adressen).

1.1.2. Online anonieme markten

De online anonieme markten, ook wel 'cryptomarkten' of 'darkmarkets' genoemd, worden beschreven als online platformen waarbij diensten en goederen verhandeld worden door aanbieders en klanten die digitale encryptie gebruiken om hun identiteit te verhullen [28]. Ze kunnen vergeleken worden met websites zoals marktplaats.nl of het Amerikaanse eBay, maar dan met een laag van privacy- en anonimiserende technieken. Het hosten van deze online anonieme markten op het darkweb zorgt

[1] Favela: Braziliaanse sloppenwijk; door misdaad geteisterde wijken, gedomineerd door criminele gangs en slechts de sporadische aanwezigheid van politie [40]



Aantal hidden services op Tor in de periode van januari 2015 t/m juni 2018[33]

ervoor dat het lokaliseren van de server en dus de handhaving door politie- of opsporingsdiensten een stuk lastiger is.

Hoewel er op de online anonieme markten een groot aantal verschillende cryptocurrencies gebruikt kunnen worden om goederen of diensten mee te betalen, wordt nog steeds met name Bitcoin gebruikt in transacties. Bitcoin en andere cryptocurrencies bieden kopers en verkopers een hogere mate van anonimiteit dan 'traditionele' betaalmiddelen zoals credit cards, PayPal of Western Union.

Een online anonieme markt is, net zoals de surfaceweb varianten zoals marktplaats.nl of eBay.com, een gedecentraliseerde platformdienst. Het platform faciliteert het aanbieden, kopen en reviewen van goederen en diensten, terwijl het aanbod op de markt zelf van verschillende autonome aanbieders komt. Een marktplaats 'bestaat' gemiddeld slechts 9 maanden: bestaan wil zeggen dat het platform beschikbaar is voor gebruikers op het darkweb. Sommige marktplaatsen worden overgenomen en gesloten door politie of opsporingsdiensten, sommige marktplaatsen eindigen in een 'exit scam' waarbij de eigenaren er vandoor gaan met het geld van kopers dat gereserveerd staat om over te maken naar aanbieders en van sommige marktplaatsen is de reden voor het verdwijnen niet bekend.

"Criminals have always tried to be one step ahead, and technology often revolutionizes how illegal goods are traded or sourced. But when drugs, guns and pedophilia moved onto the dark web, the cops came too."
Joseph Cox [12]

1.1.3. Politie- en opsporingsdiensten

In de omgeving van online anonieme markten zijn echter ook politie- en opsporingsdiensten verschenen; de handel in illegale goederen en diensten in een 'openbare ruimte' op het Tor-netwerk, gecombineerd met grote geldstromen van en naar de markten, hebben geleid tot meerdere strafonderzoeken naar kopers, verkopers en facilitators van dit soort markten. Geruchtmakende zaken waren bijvoorbeeld het onderzoek naar Ross Ulbricht, de oprichter van Silk Road 1, en operatie Onymous (FBI) en Bayonet (Team High Tech Crime Politie) waarin na het sluiten van AlphaBay de Hansa Market achter de schermen voor een maand overgenomen werd. Deze grootschalige interventies hebben hun effect gehad: de stijgende trendlijn in omzet en advertenties lijkt doorbroken te zijn door de verstoring van het criminele ecosysteem van binnen uit [26]. Hoewel de criminaliteit zich naar besloten Telegramgroepen en decentrale markten lijkt te verplaatsen [35], zijn er nog diverse soortgelijke marktplaatsen actief. Het European Cybercrime Centre van Europol heeft dit soort 'online criminal markets' dan ook aangeduid als een misdaadprioriteit voor de Europese lidstaten [20].

Uit scrapes van onderzoekers en uit inbeslaggenomen databases van strafzaken is veel data beschikbaar over het criminele fenomeen en overal binnen politie- en opsporingsdiensten worden teams opgezet om aan de hand van deze data delicten en verdachten op te sporen. Hierbij is de grootste uitdaging het vinden van die delicten of verdachten die passen bij de jurisdictie of bevoegdheden van de desbetreffende politie- of opsporingsdienst. Door de verhulling van IP-adressen op het Tor-netwerk en het gebruik van pseudoniemen is echter het vaststellen van de nationaliteit, herkomst of vestigingslocatie van kopers en verkopers zeer uitdagend. Om de omvang van de problematiek binnen de landsgrenzen in te schatten, wordt er door beleidsmakers gebruik gemaakt van meerdere grootschalige onderzoeken door wetenschappers en kennisinstituten naar de geografische verspreiding van de criminaliteit op online anonieme markten. Deze onderzoeken creëren echter slechts een algemeen beeld van de betrokkenheid op het niveau van een land en kunnen zelden de aanleiding zijn van een opsporingsonderzoek naar een concreet delict of verdachte, omdat ze niet ingaan op de persona betrokken bij de getoonde cijfers. Daarbij brengen zij slechts de vermoedelijke oorsprong in kaart van de goederen die daadwerkelijk verzonden moeten worden, zoals bijvoorbeeld drugs en wapens.

1.2. Doelstelling en onderzoeksvragen

Ook Nederlandse politie- en opsporingsdiensten zien zich geconfronteerd met een gat tussen de *intelligence*[2] beschikbaar over de algemene trends in de criminaliteit zichtbaar op dit soort online anonieme markten, en de intelligence benodigd voor het kunnen ontwerpen van een bestrijdingsstrategie op het gebied van de criminaliteit waar ze verantwoordelijk voor zijn. Ze hebben niet de beschikking over middelen en strategieën om uit verkregen data inzichtelijk te maken welk type producten of diensten er wordt aangeboden en welke hiervan enige betrokkenheid met Nederland of een Nederlander hebben. Een methode ontbreekt. De hoofdvraag die dit onderzoek tracht te beantwoorden is dan ook:

Hoe kan de geografische betrokkenheid van een land op het niveau van distinctieve productgroepen op online anonieme markten voor politie- en opsporingsdiensten bepaald worden?

Deze hoofdvraag zal beantwoord worden door de sequentiële beantwoording van de volgende subvragen:

1. Wat zijn de lacunes in de huidige wetenschappelijke literatuur op het gebied van geo-attributie op online anonieme markten?
2. Wat is het kader, waarbinnen een methode moet blijven om in de praktijk van toegevoegde waarde te zijn voor politie- en opsporingsdiensten?
3. Hoe kan er geo-attributie op productniveau op online anonieme markten uitgevoerd worden?
4. Tot welke resultaten leidt de toepassing van deze methode op een concrete casus van een politie- of opsporingsdienst?

1.3. Structuur rapport

Om politie- en opsporingsdiensten grip over hun data te geven en hen in staat te stellen richting te geven aan hun opsporingsbeleid aan de hand van verkregen of verworven data, zal dit onderzoek zich richten op het ontwerpen van een methode om uit data van online anonieme markten tot een inzicht van de Nederlandse betrokkenheid op het niveau van distinctieve productgroepen te komen. Allereerst zal er in hoofdstuk 2 onderzocht worden waarom deze vraag met de huidige beschikbare literatuur nog niet te beantwoorden is. Het nut en de toepasbaarheid van de methode zal in de praktijk getoetst worden door toepassing van de methode op de casus van de FIOD[3] in hun zoektocht naar tekenen van Nederlandse fraudeurs of facilitators op online anonieme markten. In hoofdstuk 3 zal de casus van de FIOD geïntroduceerd worden. Op basis van de theoretische inzichten uit de literatuur en de praktische kaders uit de analyse van de casus zal er in hoofdstuk 4 een methode ontworpen worden. Deze methode wordt in hoofdstuk 5 toegepast op de casus van de FIOD, om de resultaten van het gebruik van de methode in kaart te brengen. Ten slotte volgt in hoofdstuk 6 de conclusie op de hoofdvraag en aanbevelingen voor verder onderzoek naar de methode en zijn toepassing.

[2]Intelligence: "Geanalyseerde informatie en kennis, op grond waarvan beslissingen over de uitvoering van de [...]taak kunnen worden genomen" [24]

[3]FIOD: Financiële Inlichtingen en Opsporingsdienst van de Belastingdienst gericht op het bestrijden van financiële, economische en fiscale fraude [2].

2

Achtergrond en gerelateerd werk

2.1. Introductie

Sinds de officiële lancering van Tor voor het algemene publiek in 2002 en de opkomst van de verschillende 'darknets'[3] zoals Tor, I2P en Freenet, hebben veel onderzoekers het gebruik van deze netwerken voor criminele doeleindes onderzocht. Daarbij zijn de online anonieme markten sinds de oprichting van Silk Road 1 (2011) een populair onderwerp geweest van diverse onderzoeken, met name gefocust op de drugshandel zichtbaar op deze markten. Op het vlak van *advertising* is er de afgelopen jaren sterk geïnvesteerd in manieren om de geolocatie van de gebruiker van een website of applicatie te kunnen benaderen, om daarmee relevantere en persoonlijke advertenties aan te kunnen bieden. Soms is dat eenvoudig af te leiden uit de 'digitale handtekening' van het gebruikte apparaat. Aangezien dat niet altijd mogelijk is, zijn er meerdere technieken ontwikkeld om op basis van de inhoud van iemands communicatie af te leiden waar die gebruiker zich bevindt.

In dit hoofdstuk wordt er een verkenning gemaakt van de huidige status van de wetenschappelijke literatuur. Eerst zullen de inzichten uit onderzoeken naar het aanbod op online anonieme markten beschreven worden. Vervolgens zal er ingegaan worden op de methodes die ingezet worden voor de geo-attributie van de gebruikers van websites of applicaties op basis van tekstuele inhoud. Ook zal er gekeken worden naar de toepassing van geo-attributie op online anonieme markten en de toepasbaarheid van alle voorgaande inzichten voor politie- en opsporingsdiensten. Het hoofdstuk wordt afgesloten met het duiden van de gevonden lacunes in de huidige wetenschappelijke literatuur.

2.2. Het aanbod op online anonieme markten

Er zijn meerdere grootschalige onderzoeken gedaan naar het algemene aanbod op online anonieme markten. Zo hebben Soska & Christin in het eerste longitudinale onderzoek in 2015 gevonden dat ongeveer 70% van het aanbod uit drugs bestaat [37]. Dit percentage lijkt een geaccepteerde ondergrens te zijn: in 2016 kwamen Baravalle, Lopez en Lee in hun onderzoek naar Agora op basis van scrapes uit 2015 uit op 80% drugs [1]. In de analyse van EMDDA-Europol van Alphabay in 2017 (geschreven door Nicolas Christin), kwam het percentage drugs ook boven de 80% uit [8]. Dit is dan ook de reden voor Christin om te concluderen dat "the ecosystem, as a whole, appears relatively stable over time"[8], wanneer het gaat over de gevonden trends en marktaandeel van producten.

De data voor het onderzoek naar het aanbod op online anonieme markten wordt vaak verzameld aan de hand van een methode die door Thelwall, Liwen en Bjorneborn geduid is als 'web-o-metrics', namelijk de verzameling van online data voor de kwalitatieve analyse van web-gerelateerde fenomenen [38]. De data die verzameld wordt bestaat in dit geval uit gekopieerde webpagina's. Deze kopieën zijn verkregen door middel van een 'webscraper': een programma dat volgens vooraf ingevoerde instellingen een bepaalde website of gedeelte van een website kopieert. Zoals in de replicatie van het onderzoek van Dolliver [15] door Munksgaard, Demant en Branwen [31] naar voren kwam en zoals ook uitgebreid besproken wordt in Soska en Christin [37], kan er een grote bias in de data ontstaan door onvolledige scrapes. Om de compleetheid zo veel mogelijk te kunnen garanderen, is er bij het scrapen een volgordelijkheid ingebouwd. Dit betekent dat bepaalde pagina's of velden altijd als eerste

gescrapte worden, om een grotere kans te hebben dat de belangrijkste informatie ook bij onvolledige scrapes gekopieerd is.

Door de grote omvang in verkregen data uit scrapes, wordt de analyse vaak uitgevoerd met behulp van machine learning algoritmes. Machine learning algoritmes kunnen onderverdeeld worden in grofweg vier types: supervised learning, unsupervised learning, semi-supervised learning en reinfocement learning. Voor het bepalen van het aanbod, is in de onderzoeken van Soska en Christin en Van Wegberg et al. gebruik gemaakt van supervised learning, namelijk classificatie. De benodigde gelabelde data is verkregen door eigen handmatige labeling, of door het gebruikmaken van data uit scrapes die reeds gelabeld was. Met een getrainde SVM-classifier waren Soska en Christin in staat om in 97% van de listings correct te classificeren in één van hun 22 productcategorieën.

Omdat het aanbod van online anonieme markten voor het overgrote deel uit drugs bestaat, wordt het niet-drugs gedeelte van de markt vaak afgedaan in een categorie 'overig', 'anders' of 'diversen'. Dit betekent dan ook dat wanneer er uitspraken worden gedaan over de omzet van de markten in dollars of over het aantal listings per tijdseenheid, dat er eigenlijk uitspraken worden gedaan die voortkomen uit de verkopen van drugs. Dit is dan ook de reden geweest voor Van Wegberg et al. [39] om juist het niet-drugs-gerelateerde aanbod te onderzoeken, vanuit de hypothese dat dit soort online anonieme markten ook een erg logische plek zijn voor de verkoop van cybercrime componenten. Uit hun onderzoek bleek dat er inderdaad cybercrime componenten op online anonieme markten verhandeld worden en dat de omzet van de handel in cybercrime componenten in 2016 ten minste 3 miljoen dollar bedroeg. Deze omvang van de omzet gegenereerd door de handel op darkmarkets wordt bij benadering berekend aan de hand van het aantal feedback dat een listing krijgt. Het vermenigvuldigen van het aantal feedback met de prijs van de advertentie kan op die manier gebruikt worden als een ondergrens voor het aantal verkopen in een periode [9][37]. Dit soort schattingen van de omzet zijn dus een ondergrens voor de totale omzet op dit soort markten, aangezien de bestelde of verkochte kwantiteit niet uit de feedback af te leiden is.

2.3. Geo-attributie op basis van tekstuele inhoud

Het aanbieden van op locatie gebaseerde advertenties is met de komst van draagbare apparaten (mobiele telefoons, tablets, laptops) een belangrijk onderzoeksgebied voor marketing en advertisement geworden. In veel omstandigheden wordt er voor het afleiden van de locatie gebruik gemaakt van de meegezonden GPS-coördinaten, of wanneer dat niet mogelijk is, het meegezonden IP-adres. Er zijn echter ook populaire platformen, interessant voor marketeers, waarbij dit niet of slechts ten dele mogelijk is. Voor dit soort platformen zijn er methoden ontwikkeld waarbij er, op basis van de inhoud van de communicatie van een gebruiker op dat platform, een analyse wordt gemaakt die leidt tot een inschatting van de locatie van die gebruiker. Er zijn in de literatuur grofweg vier methodes van geo-attributie te vinden:

1. Analyse van de tekstuele inhoud aan de hand van vooraf bepaalde termen, uit bijvoorbeeld een zogenoemde *gazetteer* (externe database met daarin plaatsnamen, adressen en postcodes);
2. Analyse van de tekstuele inhoud met probabilistische (taal)modellen, waarbij er op basis van het gebruik van bepaalde woorden of termen de kans op een locatie bepaald wordt;
3. Deductie van de locatie aan de hand van het sociale netwerk van een gebruiker;
4. Een combinatie van bovenstaande technieken.

Een populair platform waar dit voor onderzocht is, is Twitter. Zo gebruikten Cheng, Caverlee en Lee in 2010 methode 2. om voor ieder gebruikt woord een kansverdeling te maken dat het woord een indicatie was van een bepaalde locatie. Zij noemden deze worden met een lage *spatial variation* 'woorden met een lokale geo-scope' [7]. Lokale geo-scope wordt gedefinieerd als locatie-specifieke content: specifieke plaatsnamen of specifieke woorden of uitspraken die geassocieerd kunnen worden met een bepaalde locatie. Han, Cook en Baldwin (2014) zijn hier op verder gegaan in hun onderzoek, door de zelf opgegeven locatie van Twitter-gebruikers in hun profiel als aanvullende databron te gebruiken [22]. Malmi, Solin en Gionis (2015) gebruikten methode 3. om aan de hand van de analyse van de netwerk aan de hand van een graaf-structuur de locatie van een gebruiker af te leiden [27]. Alle bovenstaande onderzoeken hebben de effectiviteit van hun methode getoetst door gebruik te maken van een (klein) percentage *ground truth* binnen hun data, of door toepassing van hun methode op een openbare dataset met bekende locaties.

2.4. Geo-attributie op online anonieme markten

Om de betrokkenheid van bepaalde regio's of landen bij de handel op online anonieme markten in kaart te brengen, zijn er ook vele onderzoeken naar de herkomst van de verkopers en aangeboden goederen uitgevoerd. Aangezien het door de gebruikte anonimiseringstechnieken van de markten op het darkweb niet mogelijk is om uit het internetverkeer of uit de website de locatie af te leiden, kunnen hier alleen methodes voor geo-attributie op basis van de tekstuele inhoud worden uitgevoerd. Dit is eigenlijk voornamelijk op één manier onderzocht, namelijk door te kijken naar het veld *ships from* of *shipping origin*. Op nagenoeg iedere marktplaats is namelijk een veld opgenomen waar een verkoper kan aangeven vanuit welk land of werelddeel hij de goederen verstuurt.

Deze manier van het benaderen van de locatie van verkopers en goederen is onder andere gebruikt door Cristin [9], Soska & Christin [37], Baravalle, Lopez en Lee [1], Broseus et al. [5], Christin [8] en Dittus et al. [14]. Doordat ongeveer 80% van de verkopers hun verzendlocatie op het platform registreert [14], is het een manier van geo-attributie die voor een groot aantal producten gedaan kan worden. Rhumorbarbe et al. stelt dat de informatie die verkopers invullen in de velden *ship from* valide is als geografische indicatoren [34], echter concluderen zij dat aan de hand van vier gedane aankopen waarbij afgeleid kon worden dat het product ook daadwerkelijk uit het opgegeven land verzonden is. Daarbij geldt natuurlijk ook dat deze velden alleen relevant zijn wanneer er een fysiek goed is dat verzonden moet worden. Bij digitale goederen of diensten is dit niet relevant, aangezien deze op een digitale manier verstuurd worden.

2.5. Toepasbaarheid voor politie- en opsporingsdiensten

Er zijn dus meerdere onderzoeken gedaan in de afgelopen jaren die het aanbod van de markten in kaart hebben gebracht en daarbij methodes voor geo-attributie hebben ingezet. Er zijn echter enkele zaken die de toepassing van deze inzichten door politie- en opsporingsdiensten bemoeilijken.

Ten eerste is er een verschil in abstractieniveau tussen de gehanteerde categorieën in de verschillende onderzoeken en het concrete niveau voor gespecialiseerde taken waar politie- of opsporingsdiensten hun informatie voor nodig hebben [13]. Zo kan een categorie 'fake documents' bijvoorbeeld nog te vaag zijn. Gaat het immers om fysieke, nagemaakte documenten, of scans van legitieme documenten? En wat voor documenten worden er aangeboden? Paspoorten, rijbewijzen, ID-kaarten, of bijvoorbeeld neppe bankstatements of neppe energierekeningen? Het verantwoordelijke departement en de handhavingsaanpak van politie- en opsporingsdiensten verschilt immers sterk tussen de verschillende type documenten.

Ten tweede verschillen de gebruikte categorieën in de diverse onderzoeken. Dit is bijvoorbeeld zichtbaar gemaakt door Dalins, Wilson en Carman toen zij naar de verschillende categorieën voor het classificeren van Tor hidden site topics keken [13]. Hoewel er in dat onderzoek naar voren kwam dat een type site op meerdere manieren geclassificeerd kon worden, is dit in de onderzoeken naar het aanbod van de markten minder het geval: de verschillende soorten drugs blijven immers de verschillende soorten drugs en worden ook als zodanig geclassificeerd. Wat wel van toepassing is, zijn de verschillen in 'granulariteit': waar bijvoorbeeld Soska & Christin de categorie 'Psychedelics' hanteren in hun grafieken, gebruiken Baravelle, Lopez en Lee de categorieën van Agora, waar bijvoorbeeld 'Edibles', 'Drugs' en 'LSD' losse categorieën zijn.

Ten derde focussen de methodes van het bepalen van de geografische betrokkenheid van een land aan de hand van *ships from* zich op het in kaart brengen van de potentiële verdachten: de verzenders van drugs. Daarbij wordt er niet gekeken naar het aanbod op online anonieme markten dat duidt op potentiële slachtoffers. Wanneer er bijvoorbeeld veel Franse gestolen creditcardgegevens aangeboden worden, kan dat een erg interessant inzicht zijn voor Franse politie- en opsporingsdiensten. De betrokkenheid van potentiële slachtoffers kan dus ook onder de geografische betrokkenheid van een land gerekend worden.

Ten vierde zijn de onderzoeken uiteraard gericht op het 'beoefenen van de wetenschap' en niet op het ondersteunen van of richting geven aan politie- of opsporingsdiensten. Soska en Christin zeggen hier dan ook expliciet over: "We certainly do not want to facilitate vendor or marketplace operator arrests. This is not just an ethical question, but is also a scientific one: our measurements, to be sound, should not impact the subject(s) being measured."[37]. Er kan echter wel een onderscheid gemaakt worden tussen de data die mogelijk leidt tot een strafonderzoek, of de inzichten en resultaten van de toepassing van een methode op de data die leiden tot het succesvol kunnen ontwerpen van een

bestrijdingsstrategie (waarbij politie- en opsporingsdiensten zelf verantwoordelijk zijn voor de data-vergaring). De gehanteerde onderzoeksmethode wordt echter, op een algemene beschrijving van de werking van het algoritme, vaak echter niet publiekelijk gemaakt. Dit betekent dat, zelfs wanneer ze zelf de data vergaren, ze de toepasbaarheid van de inzichten uit de literatuur niet kunnen inschatten aan de hand van de toepassing van de methode op hun data.

2.6. Lacunes in de huidige wetenschappelijke literatuur

Hoewel het aanbod op online anonieme markten naar geografische herkomst de afgelopen jaren door meerdere wetenschappers en kennisinstituten is onderzocht, is er nog geen methode ontwikkeld om de betrokkenheid van een land op het niveau van productgroepen te bepalen. De volgende alinea geeft antwoord op de onderzoeksvraag: *Wat zijn de lacunes in de huidige wetenschappelijke literatuur op het gebied van geo-attributie op online anonieme markten?*

Bij de gedane onderzoeken ligt er een grote focus op het aanbod van drugs, aangezien dit ongeveer 70-80% van de markt omvat. Wie naar de gehele markt zegt te kijken, kijkt eigenlijk met name naar drugs. Daarbij wordt het aanbod vaak geclassificeerd in - per onderzoek verschillende - algemene categorieën, waarbij het herkennen van de specifieke en distinctieve productgroepen in de categorie niet mogelijk is. Het bepalen van de herkomst van producten en verkopers wordt eigenlijk uitsluitend gedaan aan de hand van het zelfgerapporteerde veld *ships from*, wat niet toeziet op het gedeelte van de verkopers dat dit niet invult of de goederen die niet fysiek verzonden hoeven te worden. Methodes om op basis van de tekstuele inhoud de waarschijnlijke locatie van een gebruiker te bepalen, ontworpen voor diverse social-media platformen, worden nog niet op de gebruikers van online anonieme markten toegepast. Ten slotte zijn de onderzoeken niet gericht op het kunnen ondersteunen van politie- en opsporingsdiensten met de gedane inzichten en wordt de methode van het tot stand laten komen van het resultaat dan ook vaak niet uitgebreid toegelicht of gepubliceerd.

In dit onderzoek zal er ingaan worden op de zojuist benoemde lacunes in de literatuur en wordt er naar gestreefd om een methode te ontwerpen, waarmee de geografische betrokkenheid van een land op het niveau van productgroepen op online anonieme markten voor politie- en opsporingsdiensten bepaald kan worden.

3

Casestudy

Om het nut en de toepasbaarheid van de methode in de praktijk te toetsen, zal de ontworpen methode toegepast worden op een casus uit de praktijk. Eerst zullen de criteria voor de selectie van een casus beschreven worden, alvorens de casus van de FIOD inhoudelijk te beschrijven. Aansluitend zal de keuze voor de casus onderbouwd worden. Ten slotte zal aan de hand van de analyse van de casus een praktisch kader geformuleerd worden.

3.1. Casusselectie

Aangezien het doel is om voor politie- en opsporingsdiensten in kaart te brengen wat de geografische betrokkenheid van een land op productniveau is, zal de methode toegepast worden op een casus van een politie- en opsporingsdienst. Voor het vinden van een politie- of opsporingsdienst met een geschikte casus, zijn er meerdere criteria in acht genomen:

1. **Betrokkenheid bij het onderwerp**
   De gekozen partij moet vanuit haar werkgebied een interesse hebben in het darkweb en de online anonieme markten. Deze interesse kan bijvoorbeeld blijken uit gedane opsporingszaken of uit de toewijding van een team of mankracht op het onderwerp.

2. **Relevantie van methode**
   Het in kaart brengen van distinctieve productgroepen op een bepaald thema en de Nederlandse betrokkenheid daarvan, is relevant voor de werkzaamheden, de intelligence of de beleidskeuzes van de gekozen partij.

3. **Beschikbaarheid van data**
   Er moet data van online anonieme markten beschikbaar zijn, gerelateerd aan het thema van interesse voor de gekozen partij.

3.2. Casusbeschrijving

De gekozen casus, is de casus van de FIOD en financiële, fiscale en economische fraude op online anonieme markten. De FIOD is de opsporingsdienst van de Belastingdienst en daarmee één van de vier bijzondere opsporingsdiensten (BODs) in Nederland. Als inlichtingen en opsporingsdienst is de FIOD voor het bestrijden van financiële, economische en fiscale fraude [18]. Deze vormen van fraude zijn anders dan de vormen van fraude die de Nederlandse politie bestrijd, omdat het hier zogenoemde 'verticale fraude' betreft. Verticale fraude is fraude binnen het financiële verkeer tussen burgers en overheid (gepleegd door burgers tegen de overheid) [32].

Een overzicht van enkele veel voorkomende vormen van fraude[1] waar de FIOD tegen optreedt is [19]:

[1]Het totale overzicht van strafbare feiten waar de FIOD (mede-)verantwoordelijk voor de bestrijding is, is te vinden in het wetboek van strafrecht (WvSr) en het wetboek en de algemene wet inzake rijksbelastingen (AWR).

Fiscale thema's:

- Verborgen vermogen (buitenlandse rekeningen, rechtspersonen met vermogen en andere vermogensobjecten)
- Systeemfraude (BTW-carrouselfraude, onjuiste (suppletie) aangiften)

Financieel economische thema's:

- Faillissementsfraude
- Witwassen
- Corruptiebestrijding
- Terrorismefinanciering
- Ondermijning
- Precursoren
- Financiële fraude publieke sector
- Financiële fraude in de zorg
- Invoer, doorvoer en uitvoer van verboden middelen

De digitalisering van de criminaliteit heeft ook een effect op opsporingszaken: de afgelopen jaren wordt de FIOD meer en meer geconfronteerd met zaken met een cybercomponent. Dit betekent dat de FIOD moet blijven innoveren: wanneer de criminaliteit verandert, moeten de politie- en opsporingsdiensten immers mee veranderen. Wanneer de FIOD hier niet in slaagt, ontstaat er een onwenselijke situatie waarbij (potentiële) modus operandi, delicten en verdachten moeilijk tot niet in beeld komen [18]. Hierdoor zal de informatiepositie van de FIOD wat betreft (bekende) modus operandi, delicten en verdachten niet compleet te maken zijn, vermindert de kwaliteit en het effect van de opsporing en zouden ook de 'zware' onderzoeken (onderzoeken naar zware en georganiseerde criminaliteit) niet tot een goede afronding kunnen komen. Uiteindelijk zouden hierdoor ook de opbrengsten (het afpakken van crimineel vermogen) sterk kunnen teruglopen [18].

Aangezien de online anonieme markten nieuwe mogelijkheden bieden voor criminaliteit met een cybercomponent, hebben deze markten de interesse van de FIOD wanneer men denkt aan strategische opsporingsinformatie. Dit betekent dat de FIOD een intelligencepositie zou kunnen opbouwen op het gebied van online anonieme markten, omdat een intelligencepositie inzicht geeft in de mogelijke fraudesignalen op dat domein. Dat wil zeggen dat de FIOD inzicht verkrijgt in het aanbod op online anonieme markten, om in te kunnen schatten welk aanbod daarvan relevant is gezien de type criminaliteit waar de FIOD voor staat opgesteld: financiële, fiscale en economische fraude in Nederland[2].

3.3. Beoordeling casus

De geschiktheid van de casus van de FIOD is beoordeeld aan de hand van de in 3.1 opgestelde criteria.

1. **Betrokkenheid bij het onderwerp**
   De FIOD heeft een hoge betrokkenheid bij het onderwerp. In het verleden heeft de FIOD al meerdere opsporingszaken gerelateerd aan online anonieme markten gehad, waaronder bijvoorbeeld de zaak 'IJsberg'. Ook momenteel lopen er nog verschillende onderzoeken met een link naar online anonieme markten, zoals bijvoorbeeld de onderzoeken naar 'bitcoincashers' (FD, 2017). Daarbij heeft de FIOD sinds twee jaar een gespecialiseerd team voor Financial Advanced Cyber-gerelateerde criminaliteit.

[2] Algemene toepassing van het Nederlands strafrecht uit het Wetboek van Strafrecht, dus o.a. zaken in Nederland gepleegd, met een Nederlandse verdachte, met een Nederlands slachtoffer, volgens het universaliteitsbeginsel of wanneer de belangen van de staat ernstig worden geschaad.

2. **Relevantie van methode**
   Hoewel er delen van de organisatie van de FIOD zich bezighouden met drugs-gerelateerde criminaliteit, laat het bovenstaande lijstje aan verschillende vormen van fraude zien dat het overgrote deel niet gerelateerd is aan drugs. Bij de handel in precursoren en de invoer, doorvoer en uitvoer van verboden middelen is er een groot drugscomponent aanwezig, maar bij de overige vormen is de fraude vooral gelinkt aan diensten en handelingen en niet aan fysieke goederen. Dit betekent dat het voor de FIOD erg relevant is wanneer zij zou beschikken over het niet-drugs gerelateerde aanbod op online anonieme markten. Daarmee is de FIOD in staat om de inschatting te maken wat van dit aanbod past binnen financiële, fiscale of economisch fraude.

3. **Beschikbaarheid van data**
   Om het niet-drugs gerelateerde aanbod van online anonieme markten te kunnen analyseren, kan er gebruik worden van verschillende (al dan niet openbare) databronnen. Zo zijn er de zogeheten 'Gwern archives'[21] van een pseudoniem 'Gwern', die jarenlang op eigen initiatief verschillende online anonieme markten gescrapet heeft. Ook is het mogelijk om bijvoorbeeld data van onderzoekers te gebruiken, mochten zij bereid zijn deze ter beschikking te stellen. Informatie (in de vorm van servers, databases etc.) verkregen door middel van inbeslagname tijdens opsporingsonderzoeken zouden ook als databron gebruikt kunnen worden. Er zijn dus verschillende mogelijke databronnen, waarvan de Gwern archives openbaar en publiekelijk te gebruiken zijn.

3.4. Praktisch kader

Om de methode van toepassing te laten zijn op de casus van de FIOD, kunnen er enkele voorwaarden uit de casusbeschrijving gedestilleerd worden. Deze voorwaarden vormen samen een praktisch kader en beantwoorden de onderzoeksvraag: *Wat is het kader, waarbinnen een methode moet blijven om in de praktijk van toegevoegde waarde te zijn voor politie- en opsporingsdiensten?*

1. **Land-specifiek en variabel**
   Omdat de FIOD een Nederlandse dienst is, heeft ze bevoegdheden volgens de Nederlandse wet. Zo richt FIOD zich op zaken waar het Nederlands strafrecht uit het Wetboek van Strafrecht van op toepassing is, dus o.a. zaken in Nederland gepleegd, met een Nederlandse verdachte, met een Nederlands slachtoffer, volgens het universaliteitsbeginsel of wanneer de belangen van de staat ernstig worden geschaad. Dit betekent dat de FIOD met name geïnteresseerd is in de betrokkenheid van - zowel daders als slachtoffers - Nederlanders. Dit zal echter ook gelden voor bijv. een Italiaanse opsporingsdienst wanneer het gaat om Italianen. Dit betekent dat de methode gericht moet zijn op het in kaart brengen van de betrokkenheid van één land, waarbij het desbetreffende land aangepast kan worden aan de interesse van de desbetreffende dienst.

2. **Toepasbaarheid op eigen data**
   Hoewel er voor dit onderzoek gebruik gemaakt zal worden van data verkregen voor wetenschappelijke doeleinden, is het uiteindelijke doel dat deze methode toegepast kan worden op opsporingsdata van de politie- en opsporingsdiensten zelf. Dat betekent dat er in kaart moet worden gebracht of de gebruikte data voor de methode overeenkomt met de data beschikbaar voor politie- en opsporingsdiensten.

3. **Eenvoudig reproduceerbaar**
   Om de methode toe te kunnen passen op de eigen data, moet de methode moet van begin tot eind eenvoudig reproduceerbaar zijn.

Bij het ontwerpen van een methode zal er dus rekening gehouden worden met het kunnen variëren van het land, de toepasbaarheid van de methode op de data van politie- en opsporingsdiensten en de reproduceerbaarheid van de opgezette methode.

4

Methode

Op basis van de theoretische inzichten uit de literatuur en het praktische kader uit de analyse van de casus van de FIOD, zal er in dit hoofdstuk een methode ontworpen worden voor geo-attributie op productniveau op online anonieme markten. Als leidraad voor (het ontwerp van) de methode zal het 'Data Mining' proces van Feelders, Daniels en Holsheimer gevolgd worden [16]. Dit Data Mining proces zal eerst geïntroduceerd worden, om vervolgens de stappen in het proces uit te werken: probleemdefinitie, acquisitie van achtergrondinformatie, dataselectie, voorverwerking van data, analyse & interpretatie en rapportage & gebruik. Per stap zullen de overwegingen tijdens het vormgeven van de stap beschreven worden, om uiteindelijk een onderbouwde omschrijving te geven van de ontworpen methode.

4.1. Data Mining proces

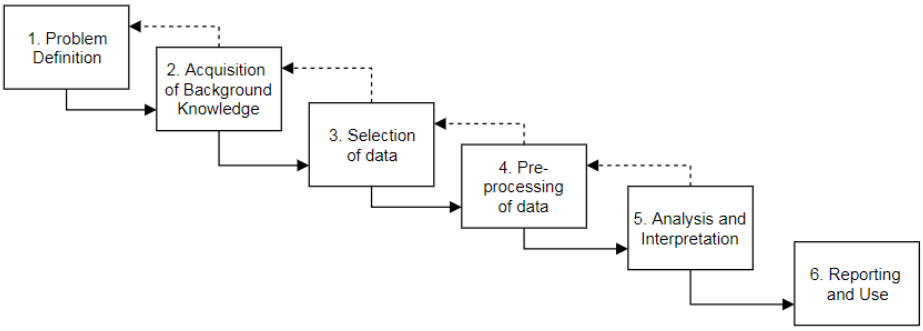
Het Data Mining proces van Feelders, Daniels en Holsheimer is gericht op het valideren van alle opeenvolgende stappen binnen data mining; een analysemethode waar vaak geen ontwerp (in de klassieke definitie van het woord binnen statistisch onderzoek) aan ten grondslag ligt. Het data mining proces begint regelmatig met data die niet verzameld is vanuit een set aan analyse-vragen, of vanuit een vooraf vastgestelde populatie, waarbij de kwaliteit van de data niet altijd voldoende is voor de analyse-doeleinden. Binnen het proces moet er dus aandacht zijn voor de uitdagingen gerelateerd aan 'niet random' samples, data-vervuiling en missende data. Het gebruik van dit proces is geschikt voor de doelstelling van dit onderzoek, om een tweetal redenen. Ten eerste past het proces goed als basis voor een methode waarbij er databronnen gebruikt worden die mogelijk onvolledig of 'vervuild' zijn, of een bias hebben. Ten tweede beschrijft het proces alle opeenvolgende stappen van probleemdefinitie tot en met rapportage, waarmee het een goede leidraad biedt voor een methode voor partijen die wellicht met dit type onderzoek niet bekend zijn, zoals politie- en opsporingsdiensten.

Feelders, Daniels en Holsheimer beschrijven de belangrijkste zes fases, namelijk de 'probleemdefinitie', 'acquisitie van achtergrondinformatie', 'dataselectie', 'voorverwerking van data', 'analyse en interpretatie' en 'rapportage en gebruik' (zie figuur 4.1 voor een visuele weergave).

In de afbeelding is zichtbaar dat er ook teruggaande of 'feedback' pijlen zijn. Deze terugkoppelingen hebben tijdens het proces plaatsgevonden; zo vereiste bijvoorbeeld de keuze voor het algoritme in de analysestap enkele extra pre-processing handelingen. Deze stappen worden echter wel opeenvolgend beschreven, waarbij informatie over verdere stappen reeds in eerdere stappen bekend wordt verondersteld.

4.2. Fase 1: Probleemdefinitie

De probleemdefinitie is nagenoeg gelijk aan de hoofdvraag: *Wat is de geografische betrokkenheid van een land op het niveau van distinctieve productgroepen op online anonieme markten?* Daarmee is de probleemdefinitie tweeledig: het onderzoek is gericht op zowel het identificeren van de productgroepen in het aanbod van online anonieme markten, als het uitvoeren van een land-specifieke geo-attributie op de listings in de gevonden productgroepen. In deze probleemdefinitie zijn meerdere termen te herkennen die eenmalig operationeel gemaakt moeten worden alvorens ze gebruikt kunnen worden,



Data mining proces volgens Feelders, Daniels en Holsheimer [16]

zoals 'productgroep identificeren' en 'geo-attributie op een listing'. Andere termen zullen voor iedere toepassing van de methode operationeel gemaakt moeten worden, zoals binnen welk thema men naar productgroepen wil kijken en voor welk land men de geografische betrokkenheid in kaart wil brengen. Deze termen zijn dan ook de input voor de volgende stap in het proces.

4.3. Fase 2: Acquisitie van achtergrondinformatie

Bij de acquisitie van achtergrondinformatie, moeten er op twee onderdelen achtergrondinformatie verkregen worden: achtergrondinformatie over de mogelijke databronnen en achtergrondinformatie over de termen uit de probleemdefinitie die operationeel gemaakt dienen te worden.

4.3.1. Achtergrondinformatie over de mogelijke databronnen

Aangezien er binnen de data mining methode vaak gebruik gemaakt wordt van databronnen die niet gemaakt zijn met statistische doeleinden in het achterhoofd, moet er extra rekening gehouden worden met mogelijke biases en selectie-effecten. Dit kan er namelijk voor zorgen dat de in de dataset gevonden patronen moeilijker te generaliseren zijn naar instanties niet in de dataset. Om dit te voorkomen, moet er voor de dataselectie bekend zijn welke mogelijke onvolkomenheden de verschillende databronnen bevatten, zodat dit mee kan wegen tijdens de selectie van data.

Voor het onderzoek naar het aanbod op online anonieme markten, zijn de mogelijke databronnen scrapes (publieke scrapes of scrapes van onderzoekers) of databases van markten verkregen door inbeslagname in opsporingsonderzoeken. Bij het gebruik van scrapes, moet er rekening gehouden worden met de impact van onvolledige scrapes. Zo is er bijvoorbeeld veel kritiek gekomen op het onderzoek van Dolliver naar het aanbod op Silk Road 2 [31]. Dolliver trok op basis van enkele crawls de conclusie dat er met name in eBooks gehandeld werd en slechts 19% van de advertenties drugs betrof, zonder hierbij acht te slaan op de reeds bekende achtergrondinformatie over de mogelijke beperkingen van het werken van scrapes/web-crawls als databronnen (zoals tevens reeds in 2.2 is beschreven).

4.3.2. Achtergrondinformatie over termen uit de probleemdefinitie

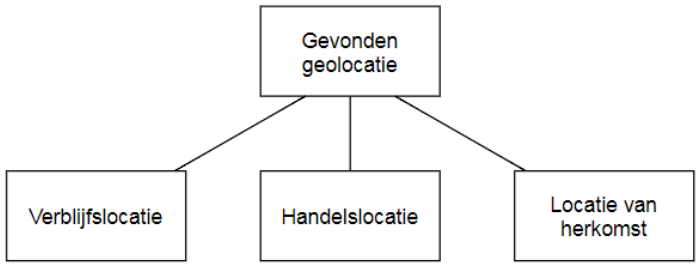
Voordat er aan de dataselectie en pre-processing begonnen kan worden, zullen alle termen uit de probleemdefinitie operationeel gemaakt moeten worden. Enkele termen zullen hieronder eenmalig concreet gemaakt worden, voor gebruik in de methode. Hiervoor zal er voortgebouwd worden op de literatuur die in hoofdstuk 2 beschreven is.

- **Productgroep identificeren**
  Met een productgroep wordt een verzameling van producten bedoeld, die aan elkaar verwant zijn doordat zij voorzien in dezelfde of vergelijkbare behoeften [23]. Het is een onderdeel van de economische terminologie die de classificatie als volgt maakt:

  I Productcategorie, bijvoorbeeld cash-out
  I.I Productgroep, bijvoorbeeld PayPal-accounts
  I.I.I Product, bijvoorbeeld een PayPal-account
  I.I.I.I. Producttype, bijvoorbeeld PayPal-accounts gekoppeld aan een creditcard

I.I.I.I.I. Productvariant, bijvoorbeeld Amerikaanse PayPal-accounts gekoppeld aan een creditcard

- **Geo-attributie op een listing**
  Geo-attributie op een listing houdt in dat er per listing bekeken wordt of er geo-attributie mogelijk is. Omdat het doel is om geo-attributie naar een specifiek land te doen, zal er ook gedefinieerd worden wat voor geolocaties er uit een listing geattribueerd kunnen worden.
  Zoals in figuur 4.2 zichtbaar is, kan de geattribueerde geolocatie eigenlijk drie verschillende types locatie zijn: de gevonden locatie kan een verblijfslocatie van een gebruiker zijn, de handelslocatie of de locatie van herkomst. Voor dit onderzoek zal hier verder geen onderscheid in gemaakt worden.



Drie mogelijke types door geo-attributie gevonden geolocaties

De overige de te concretiseren termen uit de probleemdefinitie (zie 4.2) zullen per toepassing van de methode gedefinieerd moeten worden.

4.4. Fase 3: Dataselectie

Aangezien het doel van de data mining methode is dat de data uiteindelijk 'voor zichzelf spreekt', zal bij het selecteren van de data een brede visie gehanteerd worden, die dus niet uitgaat van een vooraf opgestelde hypothese. Zoals reeds in hoofdstuk 3 beschreven is, zijn er meerdere databronnen op basis waarvan de online anonieme markten onderzocht kunnen worden. Bij de keuze tussen verschillende databronnen, zullen zaken als volledigheid en validiteit afgewogen moeten worden. Aangezien de geo-attributie op productniveau gebruikmaakt van zowel de titel en beschrijving van de listing, als de naam van de verkoper, dienen minstens deze velden in de dataset beschikbaar te zijn. Daarbij ziet deze methode niet toe op het categoriseren van het aanbod, dus dient een onderverdeling naar categorie al beschikbaar te zijn, of gedaan te zijn voordat men aan deze methode begint.

4.5. Fase 4: Pre-processing

Binnen de pre-processing van data zijn twee activiteiten te onderscheiden. Eerst wordt er een verkennende analyse uitgevoerd, om vervolgens de data op te schonen en te transformeren naar het juiste format voor verdere analyse.

4.5.1. Verkennende analyse

De verkennende analyse is bedoeld om een verder inzicht te krijgen in de dataset. Zo worden er beschrijvende statistieken gebruikt om totale aantallen, unieke waardes en frequenties te vinden. Deze aantallen worden visueel gemaakt in grafieken, om een beeld te krijgen bij de data die geanalyseerd gaat worden.

4.5.2. Opschonen en transformeren van de data

Het opschonen en transformeren van de data heeft als doel om de data zo te bewerken dat een zo hoog mogelijke kwaliteit van de analyse bereikt kan worden. Vanwege de ongestructureerdheid van 'ruwe' tekstuele data, hangt de kwaliteit van de verdere analyse af van de mate waarin de data gestructureerd kan worden. Dit is dan ook de reden dat Feldman en Sanger schrijven dat het succes van tekst mining eigenlijk gedefinieerd wordt door de uitgebreide pre-processingstechnieken [17] (in

populairdere bewoordingen: *"Garbage in = garbage out"*). De pre-processing van de data is dan vaak ook het meest tijdrovende gedeelte van het data mining proces. Welke stappen er voor de opschoning en transformatie van de data allemaal gezet dienen te worden, is naast afhankelijk van het type data, ook afhankelijk van het type algoritme dat er voor de analyse gebruikt zal worden.

Het type data waar mee gewerkt wordt is tekstuele data en het algoritme dat gebruikt zal worden is het LDA-algoritme, dus is er een voorwerkingsproces ontworpen dat de tekstuele data structureert in een format dat gebruikt kan worden in een LDA-algoritme. Vooropuitlopend op het LDA-algoritme, zal vanaf nu de term 'document' gebruikt worden voor de data van één listing die geanalyseerd wordt en de term 'corpus' voor de collectie van alle documenten (dus alle listings) in de dataset.

Er zijn vele verschillende stappen mogelijk, dus is er aan de hand van literatuur [17][41] en voorbeelden uit de praktijk [11][25] een keuze gemaakt om de volgende stappen opeenvolgend uit te voeren:

1. **Samenvoegen titel en beschrijving**
   Omdat de titel en beschrijving samen het aangeboden product beschrijven, zijn deze twee velden samengenomen voor de analyse.

2. **Parsen en opschonen**
   Het verwijderen van alle opmaak, tekens en woorden die geen toegevoegde waarde leveren aan de verdere analyse. Omdat het LDA-algoritme gebruik maakt van het bag-of-words model, is de volgorde van de woorden in de zin niet belangrijk en kunnen bijvoorbeeld alle leestekens verwijderd worden. De gebruikte parse-regels zijn als volgt:

   (a) URL's verwijderen en vervangen door 'url'
   (b) E-mailadressen verwijderen en vervangen door 'email'
   (c) PGP keys verwijderen en vervangen door 'PGPkey'
   (d) End-of-line codes verwijderen (\\r of \\n)
   (e) Alle getallen en punctuatie verwijderen
   (f) Alle woorden gelijk aan of korter dan twee letters verwijderen
   (g) Alle woorden gelijk aan of langer dan 28 letters verwijderen
   (h) Alle losse spaties tussen woorden verwijderen

3. **Verwijderen van 'stopwoorden'**
   Het LDA-algoritme werkt aan de hand van woordfrequentie, waarbij de hoge frequentie van een woord in een document gebruikt kan worden voor het bepalen van één van de thema's van het document. Bepaalde vaak gebruikte woorden, zoals lidwoorden of persoonlijke voornaamwoorden, hebben echter nauwelijks een waarde binnen het bepalen van een thema. Dit soort woorden worden 'stopwoorden' genoemd en worden dus verwijderd uit de documenten. Daarbij zijn er vaak context-specifieke woorden die gebruikt worden in het corpus, die geen betekenis hebben voor het doel van de uiteindelijke analyse. Bij online anonieme markten is het bijvoorbeeld het geval dat, aangezien het een handelsplatform is, veel verkopers dezelfde woorden gebruiken om hun 'waren' (de producten) aan te bieden, ook al zijn de producten zelf anders. Dit soort context-specifieke woorden zijn te herkennen aan het feit dat ze een hoge frequentie hebben over het gehele corpus, maar inhoudelijk weinig onderscheidend zijn voor het herkennen van de clusters.

4. **Tokanization**
   Bij tokanization wordt van ieder woord een los 'token' gemaakt. Dit stelt het LDA-algoritme in staat om aan ieder uniek woord in het corpus een aparte identifier te hangen en zo de frequentie van ieder woord in een document te bepalen.

5. **Lemmatization**
   Het lemmatizen (ook wel 'stemming' genoemd) brengt ieder woord terug naar het lemma in het woordenboek. Dit zorgt er bijvoorbeeld voor dat verschillende werkwoordsvormen allemaal voor het gebruik (de frequentie) van eenzelfde woord tellen. Omdat het kan voorkomen dat door lemmatization er alsnog stopwoorden in het corpus aanwezig blijven, worden de stopwoorden na deze stap nogmaals uit de documenten verwijderd.

Dit bovenstaande proces zou bijvoorbeeld de volgende bewerking opleveren. Hetgeen dikgedrukt is gemaakt, is wat er in de opeenvolgende stap zal wijzigen.

- Document voor voorbewerking: "***T**he quick brown fox jumps over the lazy dog**.**"
- Document na parsen en opschonen: "***the** quick brown fox jumps **over the** lazy dog*"
- Document na verwijderen van stopwoorden: **"***quick brown fox jumps lazy dog* **"**
- Document na tokanization: *"quick","brown","fox","**jumps**","lazy","dog"*
- Document na lemmatization: *"quick","brown","fox","jump","lazy","dog"*

Na het uitvoeren van deze stappen is het door de verkennende analyse duidelijk hoe de data waar mee gewerkt zal worden er uit ziet en is de data door het opschonen en transformeren klaar voor analyse.

4.6. Fase 5: Analyse en interpretatie

Om tot een goede methode voor analyse te komen, zijn er meerdere afbakeningen gemaakt. Ingegeven door het type beschikbare date en de omvang van die data, is er gekozen voor een machine learning algoritme. Omdat het de bedoeling is om inzicht te creëren waar dit er nog niet is, is er gekozen voor een unsupervised approach. Bij unsupervised learning hoeft er immers geen aanname te zijn over wat de data bevat en is het niet noodzakelijk om een gelabeld sample te hebben van de te analyseren data. Op deze manier is het mogelijk om door inductie inzichten uit de data te genereren. Binnen de unsupervised learning technieken is er gekozen voor clustering als methode om tot productgroepen te komen. Als clusteringstechniek is er gekozen voor LDA, aangezien deze techniek goed toepasbaar is op het clusteren van tekstuele data.

4.6.1. LDA-clustering naar productgroepen

LDA, oftewel Latent Dirichlet Allocation, is een techniek allereerst beschreven door David Blei [4]. Het uitgangspunt van de techniek is dat ieder document (de listing) bestaat uit een kansverdeling van allerlei verschillende topics. Zo kan een document bijvoorbeeld voor 50% bestaan uit het topic 'Cash-out', voor 40% uit het topic 'Guide' en voor 10% uit het topic 'Custom'. Daarbij bestaat ieder topic weer uit kansverdelingen over de woorden in het corpus. Een woord kan dus positieve kanswaardes hebben in enkele of alle topics. Dat wil zeggen, het woord 'account' kan bijvoorbeeld voor 4% duiden op het topic 'Account', maar ook voor 2% duiden op het topic 'Cash-out'. LDA is een generatief proces, wat wil zeggen dat het op basis van de veronderstelde kansverdelingen de toewijzingen van woorden aan topics opbouwt. Dit heeft als gevolg dat aan het begin van het algoritme, het aantal topics opgegeven dient te worden. Ook leidt dit er toe dat de kansverdelingen iedere keer dat het algoritme uitgevoerd wordt, net iets anders zullen zijn, ook bij een gelijkblijvend aantal clusters.

Aangezien het aantal gekozen topics voor de LDA clustering gelijk is aan het aantal gevonden productcategorieën, maakt dat het kiezen van het juiste aantal topics erg belangrijk. Dit lijkt eigenlijk vrij tegenstrijdig: LDA is een methode om structuren in de data in kaart te brengen die niet eenvoudig zichtbaar zijn en toch moet de belangrijkste parameter, namelijk het aantal clusters, zelf worden geschat. Er zijn daarom ook verschillende methodes ontworpen om de verschillende LDA modellen, voortkomend uit een verschillend aantal clusters, te evalueren.

Voor het toepassen van LDA kunnen verschillende tools gebruikt worden. In Python, een taal vaak gebruikt voor data-analyse, zijn bijvoorbeeld meerdere open source packages beschikbaar waarin de LDA techniek eenvoudig toe te passen is. Daarnaast is het bijvoorbeeld ook mogelijk om LDA toe te passen in R- of Matlab-omgevingen.

Evaluatie van LDA modellen

Voor het evalueren van verschillende LDA modellen, dat wil zeggen LDA modellen met een verschillend aantal clusters, zullen er twee methodes gebruikt worden. Er zijn veel verschillende methodes om de clusters van een LDA-model te evalueren, zoals bijvoorbeeld de 'purity score', sparsity of PMI (pointwise mutual information). Elk van deze methodes kent echter een technische invalshoek, waarbij vaak LDA modellen met een hoog aantal clusters tevens hoog scoorden. In de praktijk is echter

bewezen dat er een sterke relatie is tussen een hoger aantal topics en de kans dat deze topics als incoherent worden verklaart door domein-experts [29]. De in dit onderzoek gekozen methodes voor de evaluatie van de gevormde clusters richten zich juist op de coherentie van de clusters, terwijl ze ook de afstand tussen de gevormde clusters in acht nemen.

1. Coherentie score
De coherentie-score van Mimno et al. is gebaseerd op de top 15 woorden van een topic en kijkt naar hoe vaak paren van deze 15 woorden samen voorkomen in het corpus. De aanname is dat wanneer deze paren vaak samen voorkomen, ze een coherente betekenis hebben omdat ze elkaar 'ondersteunen' in de documenten [29]. De coherentie score is op deze manier ook een graadmeter voor de kwaliteit van de clusters, waarbij geldt 'hoe hoger hoe beter'. In hun onderzoek leiden scores hoger dan 0.5 vaak tot clusters van een hoge kwaliteit, wanneer deze kwaliteit beoordeeld werd door een domein-expert.

2. Visualisatie van topics
Visualisatie van de topics wordt uitgevoerd aan de hand van de tool 'LDAvis', ontworpen door Sievert en Shirley [36]. Per topic worden de top-30 meest *relevante* woorden getoond, waarbij relevantie berekend wordt aan de hand van zowel de woord-topic specifieke kansverdeling, als aan de hand van het aantal keer dat het woord voorkomt in het totale corpus (oftewel, de marginale kansverdeling in alle documenten van het corpus). De aanname is dat wanneer een woord minder vaak voorkomt in het corpus, het dus van grotere waarde is voor de topics waartoe het behoort. Dit samen wordt 'relevance' genoemd en is zichtbaar als λ op de plots. De afstand tussen de clusters wordt berekend aan de hand van de overeenkomsten in de woorddistributie. De omvang van de clusters geeft aan hoe vaak het topic voorkomt in de totale corpus.

Afweging tussen aantal clusters aan de hand van coherentie score en visualisatie
Allereerst kan er door het toepassen van een variërend, oplopend aantal clusters aan de hand van het coherence measure een eerste inschatting gemaakt worden van het aantal clusters dat leidt tot een goed LDA-model. Zo kan bijvoorbeeld de coherence-measure van een model met 2, 5, 8, ..., 25 clusters berekend worden. Wanneer de resultaten niet een eenduidige piek laten zien, kan de analyse worden herhaald met een kleiner bereik aan clusterwaardes, zoals bijvoorbeeld 2 t/m 10. Vervolgens kunnen de beste modellen of het beste model worden gevisualiseerd met behulp van LDAvis. Wanneer men nog twijfelt tussen twee verschillende aantal topics die wat betreft coherence measure erg dicht bij elkaar liggen, kan er op basis van de visualisatie bepaald worden welke van de twee modellen de meest eenduidige clusters bevat.

Interpretatie van resultaten van LDA-clustering

Voordat de resultaten van de LDA-clustering gerapporteerd kunnen worden, moeten ze door domein-experts geïnterpreteerd worden. Het resultaat van een LDA-clustering is immers niet een lijst van topics, maar een lijst van clusters met daarbij de woorden die met de hoogste waarschijnlijkheid een indicatie zijn van dat topic. Dit betekent dat er door experts nog een titel aan het cluster gehangen moet worden, wat dan de naam van het topic wordt.

4.6.2. Geo-attributie op basis van tekstuele inhoud

Zoals reeds in hoofdstuk 2 is beschreven, zijn er meerdere manieren voor geo-attributie voor platformen zoals Twitter ontworpen. Een van deze manieren, namelijk het analyseren van de tekstuele inhoud aan de hand van vooraf bepaalde termen, zal op de data van online anonieme markten toegepast worden. De data van online anonieme markten is qua soort en omvang van de data gelijkend aan Twitter-berichten: tekstuele data van ongeveer 100-140 woorden gemiddeld per bericht.

De termen die gebruikt zullen worden, zijn de plaatsnamen van de tien steden met het grootste aantal inwoners en de naam van het land, geschreven op alle populaire manieren. Deze termen zijn gekozen vanwege het ontbreken van de aanwezigheid van eenduidige en recente Gazetteers voor de Europese landen. Om ook rekening te houden met typefouten of bewuste mis-spellingen, zal het zoeken naar deze termen gedaan worden aan de hand van de onbewerkte data in de velden van titel, beschrijving en verkoper.

4.7. Fase 6: Rapportage en gebruik

Na de uitgevoerde analyse en de interpretatie van de resultaten, kunnen de resultaten gerapporteerd en gebruikt worden. Het beoogde gebruik is dat door politie- en opsporingsdiensten voor de uitvoering van hun werkzaamheden, de opbouw van een intelligencepositie of voor een beleidskeuzes aangaande de handhavingsstrategie van online anonieme markten. Voor de rapportage zijn er geen voorgeschreven stappen.

4.8. Methode voor geo-attributie op productniveau op online anonieme markten

In voorgaande paragrafen zijn de verschillende stappen in het Data Mining proces uitgewerkt. Het gebruik van het Data Mining proces biedt een fundering waarbij de validiteit van de opeenvolging van de stappen in de ontworpen methode gewaarborgd is. Op basis van deze stappen kan nu dus de methode beschreven worden en de volgende onderzoeksvraag beantwoord worden: *Hoe kan er geo-attributie op productniveau op online anonieme markten uitgevoerd worden?*:

1. **Probleemdefinitie opstellen**; breng in kaart binnen welke productcategorieën de productgroepen geanalyseerd moeten worden en voor welk land de geografische betrokkenheid in kaart moet worden gebracht.
2. **Achtergrondinformatie verzamelen**; verzamel achtergrondinformatie over de te onderzoeken productcategorieën en de mogelijke databronnen.
3. **Data selecteren**; kies een geschikte databron voor analyse.
4. **Pre-processing uitvoeren**; maak de data geschikt voor LDA-clustering.
5. **Analyse uitvoeren en resultaten interpreteren**; voer de clustering naar productgroepen uit aan de hand van LDA, voer geo-attributie uit aan de hand van een Gazetteer.
6. **Rapportage en gebruik van resultaten**; rapporteer de resultaten zodat ze gebruikt kunnen worden in de organisatie.

5

Toepassing methode op casus

De ontworpen methode voor geo-attributie op productniveau zal toegepast worden op de casus van FIOD, om de praktische toepasbaarheid en de gevonden resultaten te beoordelen. In de uitwerking van de methode zullen alle stappen zoals beschreven in hoofdstuk 4 gevolgd worden.

De toepassing van de methode is uitgevoerd aan de hand de Python omgeving 'Jupyter Notebook'. Deze is in zijn volledigheid te vinden de appendix (A en B).

5.1. Fase 1: Probleemdefinitie

De probleemdefinitie voor de FIOD is als volgt: Wat is de geografische betrokkenheid van Nederland op het niveau van de productgroepen mogelijk gerelateerd aan financiële, fiscale of economische fraude op online anonieme markten? De term die tijdens de acquisitie van achtergrondinformatie concreet gemaakt dient te worden, is 'de productgroepen mogelijk gerelateerd aan financiële, fiscale of economische fraude'.

5.2. Fase 2: Acquisitie van achtergrondinformatie

Voordat er aan de dataselectie begonnen kan worden, zal er achtergrondinformatie aangaande de mogelijke databronnen verkregen worden, als ook aangaande de term 'de productgroepen mogelijk gerelateerd aan financiële, fiscale of economische fraude'.

5.2.1. Productgroepen mogelijk gerelateerd aan financiële, fiscale of economische fraude

Zoals in de beschrijving van de casus in hoofdstuk 3 opgemerkt is, richt de meerderheid van de soorten fraude zich op financiële transacties en weinig op de fysieke handel in drugs. Productgroepen mogelijk gerelateerd aan financiële, fiscale of economische fraude, kunnen dan ook gedefinieerd worden als productgroepen met een financieel component. Binnen de omgeving van de online anonieme markten zal dan ook de afbakening naar het niet-drugs zijnde aanbod van de markt gemaakt worden.

5.2.2. Achtergrondinformatie mogelijke databronnen

Er zijn verschillende open source databronnen van darkmarkets te vinden, zoals bijvoorbeeld de eerder genoemde archieven van Gwern [21]. Deze databronnen zijn echter nog ongestructureerd en onbewerkt en zullen daarom eerst naar een bewerkbaar database-format met gestructureerde rijen en kolommen moeten worden getransformeerd. Deze data zal dan eerst gevalideerd moeten worden, door bijvoorbeeld statistische toetsen en externe validatie. Daarna zal er een onderscheid in de drugs en niet-drugs gerelateerde advertenties gemaakt moeten worden, gevolgd door een classificatie naar productcategorieën.

Op aanvraag kunnen vaak ook de datasets van onderzoekers verkregen worden. Zo roepen Soska & Christin in hun artikel van 2015 op om onderzoeksdata te delen, onder andere voor de validatie van de uitkomsten door replicatie van het onderzoek [37]. Voor dit onderzoek zou er, op verzoek, gebruik gemaakt kunnen worden van de data van Van Wegberg et al.. Deze dataset is een deel van

de grotere dataset van Soska & Christin, die hun dataset op verschillende wijzen gevalideerd hebben op compleetheid en representativiteit. Uit deze grote dataset hebben Van Wegberg et al. de categorieën 'Digital Goods' en 'Miscellaneous' gepakt, om daarmee effectief een afbakening te maken naar de diensten en goederen in de dataset die geen drugs zijn. Binnen deze dataset is er een verdere onderverdeling gemaakt naar cybercrime componenten als productcategorieën.

5.3. Fase 3: Dataselectie

Er is gekozen om de dataset van Van Wegberg et al. te gebruiken. De open source bronnen bevatten nagenoeg dezelfde type data (dezelfde velden en data types) als de dataset van Van Wegberg et al. en deze data is reeds gevalideerd en geclassificeerd, wat veel tijd scheelt in de pre-processing fase. Daarbij is de dataset van Van Wegberg et al. een afbakening naar het aanbod op online anonieme markten wat niet-drugs gerelateerd is.

Tabel 1 en 2 geven de structuur van de dataset aan, aan de hand van de namen van de kolommen en het soort data in de kolom. Tabel 1 betreft alle data over de listings, tabel 2 betreft de data over de feedback op de listings.

Listings

Naam kolom	*Type data*
Hash	Hashwaarde
Marketplace	Tekst-waarde
Title	Tekst-waarde
Description	Tekst-waarde
Vendor	Tekst-waarde
Vendor hash	Hashwaarde
Ships to	Tekst-waarde
Ships from	Tekst-waarde
Total sales	Decimaal getal
First observed	Datum
Last observed	Datum
Sales	Data
Prediction	Tekst-waarde
Prediction number	Getal
Category	Tekst-waarde
Source	Source file location

Feedback

Naam kolom	*Type data*
Hash	Hashwaarde
Marketplace	Tekst-waarde
Date	Datum
Giver (buyer)	Tekst-waarde
Receiver (vendor)	Tekst-waarde
Message	Tekst-waarde
Order title	Tekst-waarde
Item hash	Hashwaarde
Feedback value	Getal
Order amount	Decimaal getal
Order amount USD	Decimaal getal
Source	Source file location

Zoals zichtbaar is in bovenstaande tabellen, bestaat de dataset voor het overgrote gedeelte uit tekstwaardes en bevat de dataset de velden 'titel' (title), 'beschrijving' (description) en 'verkoper' (vendor). Dit is in overeenstemming met de benodigde data en velden voor de verdere analyse.

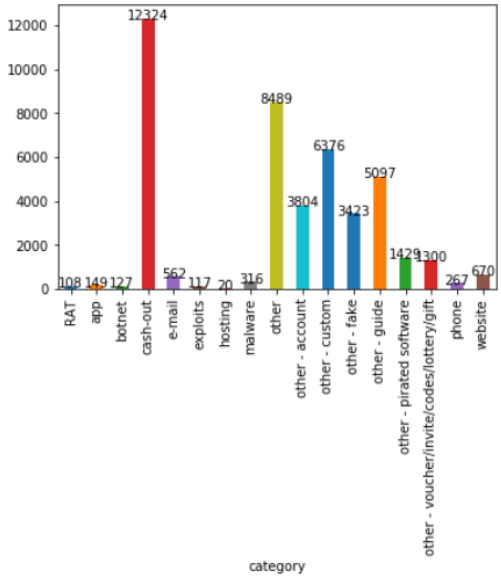
5.4. Fase 4: Pre-processing

Om de data voor te bereiden voor de analyse, zal er eerst een verkennende analyse worden uitgevoerd. Vervolgens zal de data opgeschoond en getransformeerd worden voor analyse.

5.4.1. Verkennende analyse van de dataset

Voor de verkennende analyse is er gekeken naar het aantal listings per categorie, het aantal listings per marktplaats, het aantal unieke verkopers en aantal listing per verkoper. Ook is de validiteit van de data onderzocht door te kijken naar eventuele mislukte scrapes en verkeerd geclassificeerde listings.

Aantal listings per categorie



Aantal listings per categorie

Er zijn 17 verschillende categorieën, waarbij 'cashout' (12.324 listings) en 'other' (8.489 listings) de grootste categorieën zijn. Er zijn 13 categorieën (76%) die minder dan 4457 listings (<10% van het totaal aantal listings) in hun categorie hebben en 7 categorieën (41%) die zelfs minder dan 445 listings (<1% van het totaal aantal listings) hebben. Oftewel: 90% van alle listings valt in de categorieën 'cash-out', 'other', 'other - custom' en 'other - guide'.

Aantal listings per marktplaats

Net iets minder dan de helft van alle listings (48%) is afkomstig van de Alphabay market.

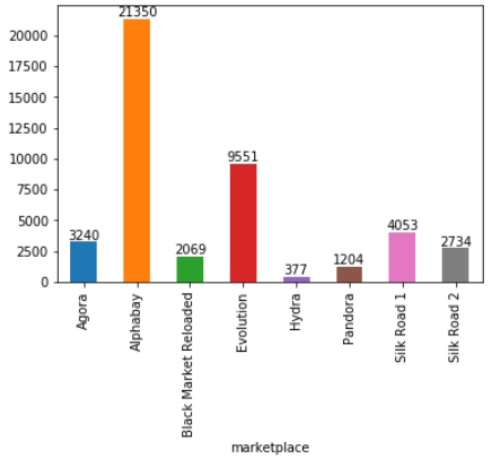
Het aantal unieke verkopers en aantal listing per verkoper

Er worden over de verschillende marktplaatsen 5585 verschillende vendornames gebruikt. De vendorname 'fake' wordt bijvoorbeeld 1332 keer gebruikt en de naam captainpicard 558 keer. Omdat de namen niet marktplaats-specifiek zijn, kan het zo zijn dat eenzelfde naam (bijv. 'DutchMasters') op drie verschillende marktplaatsen wordt gebruikt, maar hier niet dezelfde persoon achter zit. Zo worden soms 'betrouwbare namen' die gebruikt worden op een bepaald platform door andere criminelen gebruikt op een ander platform. Om er dus achter te komen hoe veel unieke verkopers er zijn, kan er beter gekeken worden naar vendor_hash: een hash van de vendor die marktplaatsspecifiek is. Hieruit blijkt dat er 6247 marktplaatsspecifieke verkopers zijn.

Er zijn 130 verkopers die meer dan 50 listings op eenzelfde marktplaats hebben geplaatst. Daarbij zijn er 2320 verkopers die slechts 1 listing hebben geplaatst.

Validatie: mislukte scrapes

Uit een eerste verkenning van de dataset, is gebleken dat er een scrapes van Silk Road 2 mislukt zijn, waardoor de beschrijving bestaat uit de gemiddelde klantwaarderingen:



Aantal listings per marktplaats

	title	description
249	asus-17-3-inch-gaming-laptop-geforce-gtx-880m-...	30 day average: 4.94\n 60 day average: 4.95...
365	survive	Overall average: 4.99
375	every-silk-road-s-money-making-methods-guides-...	30 day average: 4.91\n 60 day average: 4.91...
388	100-x-red-apples-new-stock	30 day average: 4.86\n 60 day average: 4.90...
409	1-troy-ounce-gold-coin-american-eagle-bullion	30 day average: 5.00\n 60 day average: 4.83...

Mislukte scrapes

De data van Silk Road 2 bestaat uit 195 snapshots, en listings met dit soort beschrijvingen zijn specifiek terug te leiden naar de snapshots gemaakt na 30-08-2014. Het gaat om totaal 1745 listings waarbij de beschrijving de gemiddelde klantwaardering bevat. De beschrijvingen van deze listings zullen daarom in de opschoningen en transformatie leeg gemaakt worden, aangezien deze beschrijving niks zegt over de bijbehorende productgroep.

Validatie: drugs in de 'Digital Goods' en 'Miscellaneous'

Aangezien de classifier gebruikt door Soska en Christin een recall heeft van 97%, zullen er enkele verkeerd geclassificeerde listings in deze categorieën zitten. Voorbeelden hiervan zijn dan ook te vinden door te zoeken op 'mg', 'gram', of de naam van een type drugs in de titel of beschrijving.

	title	description
1023	custom for penguin	10g mdma, 100 hits lsd
1143	Red Toyota - 200mg - 10 pc	Strong MDMA pills from EU.
1152	*Custom listing 1 for user: princeofpeace	This custom listing contains:\r3x 30 oxycotin ...
1185	10 7.5/325mg Vicoden	10 7.5/325mg Vicoden\n\n0.12953591 BTC\n\n\r\n ...
1262	IPH Isopropylphenidate {1gram} GREAT for Study...	We ship from the USA, FAST! Yes! We Accept Int...

Drugslistings in 'Digital Goods' en 'Miscellaneous'

Deze listings die deze termen bevatten, zijn echter niet allemaal drugs. De listings betreffen bijv. ook materiaal om drugs mee te verwerken, zoals bijvoorbeeld erg nauwkeurige weegschalen of custom listings. Ook is het gebruik van deze woorden een indicatie van STO (seach term optimalization zoals beschreven in Van Wegberg, 2018):

	title	description
4	rc-starter-kit-milligram-scale-and-tiny-spoons...	Highly accurate milligram (0.001g resolution) ...
101	★ eGift Cards ★ Neiman Marcus — 60% OFF ★ Doll...	PayPal checkout: https://stocard.co/\r These e...
219	forged-social-security-card	This is a listing for a forged social security...
270	custom for street	200 tabs high 90s lsd\r We aim for quick deli...
278	FULL ESCROW 10 x JUMBO the successor of tomorr...	!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!We Don't Ship ...

Search term optimalization in listings

5.4.2. Opschonen en transformeren van de data

Allereerst zullen de beschrijvingen van de listings van onjuiste scrapes van Silk Road 2 opgeschoond worden. Vervolgens zal de data volgens de in de methode (H4) beschreven stappen getransformeerd worden.

1. **Opschonen onjuiste scrapes**
   De beschrijving van de scrapes met de gemiddelde klantwaardering als beschrijving zijn verwijderd en vervangen door een spatie.
2. **Samenvoegen titel en beschrijving**
   De titel en beschrijving zijn samengevoegd tot één veld per listing.
3. **Parsen en opschonen**
   Het parsen en opschonen is gedaan aan de hand van zogeheten regular expressions die bepaalde patronen in een tekst kunnen herkennen en vervangen.
4. **Verwijderen van 'stopwoorden'**
   Dit wordt gedaan op basis van de 318 stopwoorden die opgesteld zijn door de "Glasgow Information Retrieval Group". Deze woorden zitten in het package van SK-learn. Aan deze lijst zijn nog de volgende context-specifieke woorden toegevoegd aan de hand van een analyse van de woordfrequentie en de inhoudelijke waarde van deze woorden:
   'john', 'david', 'please', 'positive', 'good', 'perfect', 'friendly', 'happy', 'great', 'provide', 'fast', 'anon', 'nice', 'thanks', 'regard', 'listing', 'thank', 'bring', 'free', 'seller', 'send', 'buy', 'need', 'use', 'check', 'know', 'want', 'make', 'do', 'don', 'sell', 'feedback', 'buyer', 'deal', 'day', 'ago', 'quick', 'answer', 'product', 'quality', 'high', 'cheap'.
5. **Tokanization**
   De tokanization is uitgevoerd met behulp van de NLTK pipeline van het Spacy package.
6. **Lemmatization**
   Elk woord is teruggebracht naar zijn of haar 'lemma', volgens de Engelse woordenlijst van nltk.stem.wordnet. De input blijft onveranderd wanneer het woord niet gevonden kan worden in WordNet.

5.5. Fase 5: Analyse en interpretatie

Nu de data klaar is voor analyse, kan het LDA-clustering algoritme en de geo-attributie op basis van tekstuele inhoud toegepast worden.

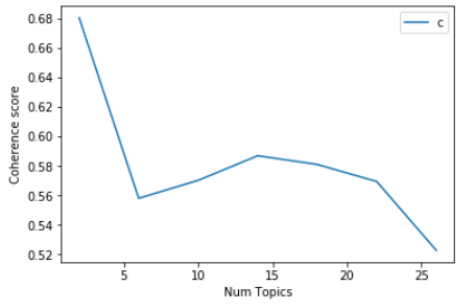
5.5.1. LDA-clustering naar productgroepen

Aangezien er voor iedere productcategorie clusters van productgroepen gevonden moeten worden, is de analyse voor iedere van de 17 in de dataset aanwezige categorieën uitgevoerd. In dit hoofdstuk zal het proces aan de hand van de analyse van het grootste cluster, namelijk 'cash-out', inzichtelijk gemaakt worden. De analyse-stappen voor alle overige categorieën en de bijbehorende resultaten zijn te vinden in Appendix B.

De LDA-clustering is uitgevoerd met het Gensim package, omdat dit type modellen eenvoudig gebruikt kan worden voor de berekening van de coherence score en als input voor de pyLDAvis visualisatie-tool.

Evaluatie van LDA modellen

Allereerst worden er LDA-modellen gemaakt met een variërend aantal clusters. Om mee te beginnen worden er modellen gemaakt met 2, 6, 10, 14, 18, 22, 26 en 30 clusters.



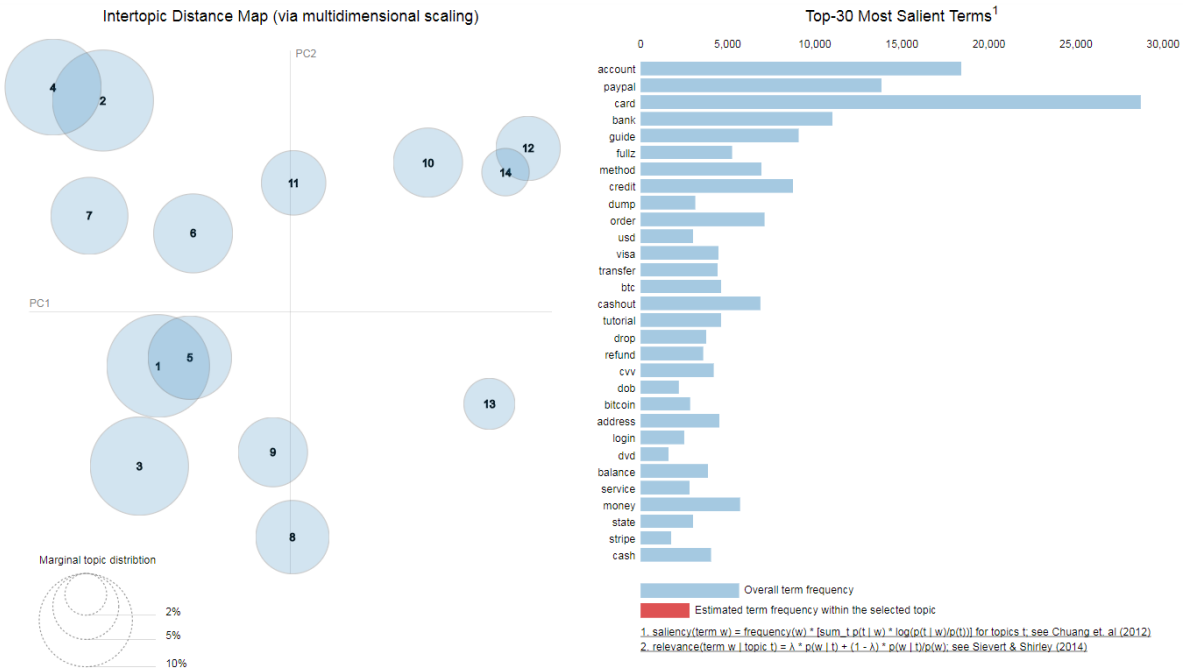
Grafiek van coherence scores voor modellen met verschillende clusters: 2 t/m 30

Vervolgens worden de exacte waardes van de coherence values weergegeven.

```
Num Topics = 2  has Coherence Value of 0.6804
Num Topics = 6  has Coherence Value of 0.558
Num Topics = 10  has Coherence Value of 0.5702
Num Topics = 14  has Coherence Value of 0.5869
Num Topics = 18  has Coherence Value of 0.581
Num Topics = 22  has Coherence Value of 0.5694
Num Topics = 26  has Coherence Value of 0.5228
```

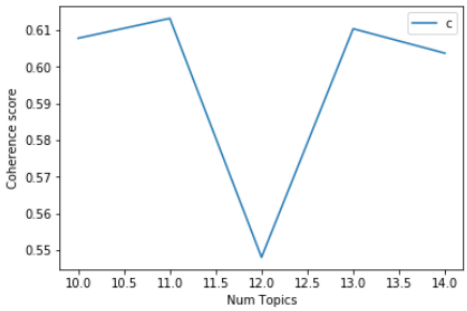
Waardes van coherence scores voor modellen met verschillende clusters: 2 t/m 30

Aan de hand van deze scores wordt er gekozen voor een model met een aantal clusters, om hiervan de clusters te visualiseren. Door middel van LDAvis kan de mogelijke overlap tussen woorden inzichtelijk gemaakt worden aan de hand van de zogeheten 'relevance'. Dit ziet er als volgt uit:



Visualisatie van LDA model van cash-out met 14 topics

Uit deze visualisatie blijkt dat er enige overlap is tussen de clusters 1 & 5, 2 & 4 en 14 & 12. Het onderzoeken of wellicht 11 clusters een betere fit voor de data is, is dus interessant om uit te zoeken. Er worden dus opnieuw LDA-modellen gemaakt, en nu met een variërend aantal clusters van 10 t/m 15.



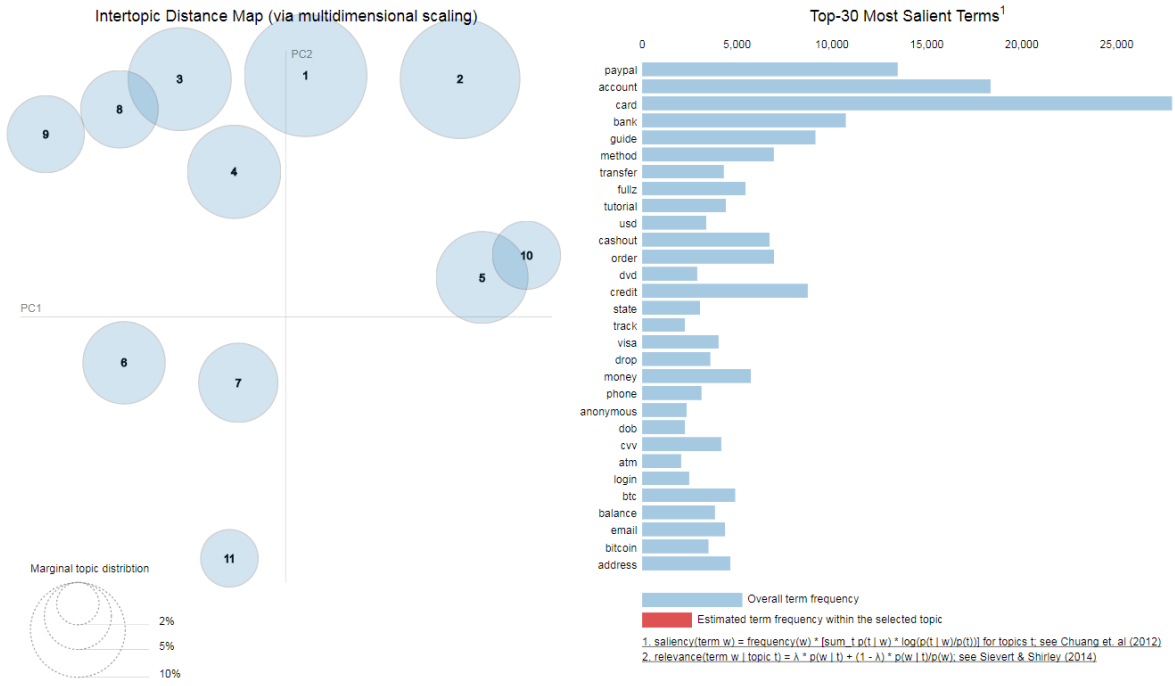
Grafiek van coherence scores voor modellen met verschillende clusters: 10 t/m 15

De bijbehorende waardes zijn nu:

```
Num Topics = 10  has Coherence Value of 0.6078
Num Topics = 11  has Coherence Value of 0.6132
Num Topics = 12  has Coherence Value of 0.5481
Num Topics = 13  has Coherence Value of 0.6104
Num Topics = 14  has Coherence Value of 0.6038
```

Waardes van coherence scores voor modellen met verschillende clusters: 10 t/m 15

Uit deze waardes blijkt inderdaad dat 11 topics een iets betere coherence value geeft. Ten slotte wordt het model met 11 topics ook visueel gemaakt met LDAvis.



Visualisatie van LDA model van cash-out van 11 topics

Uit de visualisatie blijkt dat er nog steeds overlappende clusters zijn, al is de omvang van de overlap kleiner dan bij 14 clusters. Het model met 11 clusters zal nu dus gebruikt worden voor interpretatie van de topics.

Interpretatie van resultaten van LDA-clustering

Het interpreteren van de gevonden clusters, om uiteindelijk namen aan de verschillende gevonden topics te geven, kan gedaan worden met behulp van de 10 meest voorkomende woorden van dat cluster en de top-10 listings die voor het overgrote deel uit dat topic bestaan.

Voor het de gevonden clusters in cash-out, zien bijvoorbeeld de woorden per cluster er als volgt uit:

Cluster	*10 meest voorkomende woorden*
1	method, guide, step, work, money, new, es, learn, cashout, update
2	card, tutorial, guide, cashout, method, credit, money, cash, amazon, cvv
3	btc, order, vendor, service, include, price, time, report, credit, just
4	fullz, card, information, number, address, store, canada, mmn, dob, ip
5	card, order, visa, credit, cvv, email, replacement, usa, refund, balance
6	paypal, transfer, account, cashout, money, guide, fund, ebay, verify, clean
7	account, bank, drop, login, balance, attach, password, hack, access, verify
8	usd, dvd, track, anonymous, atm, card, mbeuroteenmovs, iban, plastic, eur
9	rsv, carder, electron, level, lock, shit, teen, plc, simply, thousand
10	bitcoin, sharecash, software, hack, dude, cyber, tool, ebook, pdf, change
11	state, phone, ssn, previous, employment, dob, fullz, credit, city, info

Per cluster kan er vervolgens gekeken worden naar de top-10 van de listings en het percentage dat het topic voorkomt in elk van die listings. Bijvoorbeeld:

0,8154	★Hacked SunTrust Bank Account Logins $30K-$150K+ Acct & Routing Numbers★ Fresh I bring you freshl
0,7925	[gold] 1 Verified Business US PayPal account + 1 Bank Account attached + Very Aged (Owner joined in 200
0,7747	Wells Fargo Bank Account + Full Account / Routing numbers ++++++++++++++++++++++++++++++++++++
0,7667	****** High Balance Hacked Wells Fargo Bank Logins ******* PROMO I am selling hacked Wells Fargo lo
0,7646	Charles Schwab Bank Drops, Only$ 45 Cheapest on Alphabay! Since im new, for a limited time these bank
0,7509	✪ PREMIUM- Suntrust Bank Drops ✪ Google Voice Acc.✪ The best Price on Alpha ✪WITH VPS/RDP ✪
0,747	Huntington Bank Account Login Details -Freshly hacked Huntington bank account login details Example Ac
0,7453	Capital One 360 Bank Drops,$25 650+ Fico Scores and private SOCKS5 Comes with instruction on keeping
0,7453	✪ PREMIUM- Charles Schwab Bank Drops ✪ Google Voice Acc.✪ The best Price on Alpha ✪WITH VPS
0,7451	✪ PREMIUM- Fidelity Bank Drops ✪ Google Voice Acc.✪ The best Price on Alpha ✪WITH VPS/RDP ✪

Voorbeeld van de 10 listings met het hoogste percentage van het topic behorend bij cluster 7

Aan de hand hiervan wordt het duidelijk dat het hier om de productgroep 'Bank drops' gaat: bank accounts geopend met frauduleuze gegevens, of accounts van (al dan niet onwetende, gehackte) money mules.

5.5.2. Geo-attributie op basis van tekstuele inhoud

Zoals beschreven is in hoofdstuk 4, zullen naam van het land en de plaatsnamen van de tien grootste steden [6] gebruikt worden als Gazetteer om te zoeken in de data. Aangezien er gezocht zal worden in de ruwe, onbewerkte data, is er rekening gehouden met (al dan niet opzettelijke) schrijf en typefouten. Dit resulteert in de volgende lijst:

- nederlands
- Nederlands
- Nederland
- nederland
- Netherlands
- netherlands
- netherland
- Netherland
- Holland
- holland
- dutch
- Dutch
- Amsterdam
- amsterdam
- DenHaag

- Denhaag
- denhaag
- Rotterdam
- rotterdam
- roterdam
- Utrecht
- utrecht
- utregt
- Eindhoven
- eindhoven
- eindhove
- Tilburg
- tilburg
- Groningen
- groningen
- Almere
- almere
- Breda
- breda
- Nijmegen
- nijmegen

Het in de dataset zoeken aan de hand van deze woorden, levert 1172 resultaten op. Wanneer het wordt toegepast op bijvoorbeeld het bij 5.5.1. gebruikte topic van 'bank drops', is hier zichtbaar dat de meeste treffers komen van een verkopers die 'dutch' in hun naam gebruiken. In enkele keren gaat het om Nederlandse geverifieerde bank of paypal accounts.

5.6. Fase 6: Rapportage en gebruik

Voor de rapportage over de gevonden resultaten is er gebruik gemaakt van een presentatie intern bij de FIOD. Tevens zal de samenvatting van dit onderzoeksverslag gebruikt worden ter verspreiding en kennisdeling binnen de FIOD.

5.7. Uitkomsten toepassing methode

Aan de hand van de toepassing van de methode op de casus van de FIOD, kan de volgende onderzoeksvraag beantwoord worden: *Tot welke resultaten leidt de toepassing van deze methode op een concrete casus van een politie- of opsporingsdienst?*

De FIOD was, vanuit haar taakstelling van het bestrijden van financiële, fiscale en economische fraude en de stijgende omvang van fraude met een cybercomponent, geïnteresseerd in de relatie tussen fraude en online anonieme markten. Wat is er voor aanbod op online, anonieme markten, dat relevant is voor de FIOD?

Aan de hand van de toepassing van de methode voor geografische betrokkenheid van Nederland op productgroepen op 8 online anonieme markten in de periode van 2011-2017, heeft de FIOD inzicht verkregen in de verschillende productgroepen in het aanbod en de mogelijke betrokkenheid van Nederlandse verdachten of Nederlandse slachtoffers.

6

Conclusie en aanbevelingen

6.1. Beantwoording van de hoofdvraag

De hoofdvraag van dit onderzoek was als volgt:

Hoe kan de Nederlandse betrokkenheid op het niveau van distinctieve dienst- en productgroepen op online anonieme markten bepaald worden?

Deze hoofdvraag kan beantwoord worden aan de hand van de beantwoording van de volgende subvragen:

1. **Wat zijn de lacunes in de huidige wetenschappelijke literatuur op het gebied van online anonieme markten?**
   Het huidige onderzoek naar online anonieme markten heeft zich met name gefocust op het in kaart brengen van het aanbod van drugs, aangezien dit ongeveer 70-80% van de markt omvat. Het bepalen van de herkomst van producten en verkopers wordt eigenlijk uitsluitend gedaan aan de hand van het zelfgerapporteerde veld *ships from*, wat niet toeziet op het gedeelte van de verkopers dat dit niet invult of de goederen die niet fysiek verzonden hoeven te worden.

2. **Wat zijn de praktische kaders waarbinnen een methode ontworpen moet worden?**
   De ontworpen methode moet toegepast kunnen worden op ieder land, op verschillende databronnen van online anonieme markten en de methode moet reproduceerbaar zijn voor politie- en opsporingsdiensten.

3. **Met welke methode kan de Nederlandse betrokkenheid op het niveau van distinctieve dienst- en productgroepen op online anonieme markten bepaald worden?**
   Aan de hand van het Data Mining proces is er een methode ontwikkelt die van probleemdefinitie tot en met rapportage en gebruik voorschrijft hoe, aan de hand van de data van online anonieme markten, de Nederlandse betrokkenheid door middel van geo-attributie op het niveau van productgroepen bepaald kan worden. De productgroepen worden verkregen door clustering aan de hand van de LDA-techniek. De geo-attributie wordt uitgevoerd aan de hand van een zelf opgestelde Gazetteer van de tien grootste steden van een land, als ook de naam van het land zelf.

4. **Tot welke resultaten leidt dit, wanneer deze methode wordt toegepast op een concrete casus?**
   Het toepassen van deze methode resulteert in een overzicht van de productgroepen per categorie en het aantal gevonden Nederlandse attributies per productgroep. Voor verdere analyse kunnen ook alle listings in een productgroep, of alleen de listings met een Nederlandse attributie opgevraagd worden.

Daarmee kan de hoofdvraag als volgt beantwoord worden:

De Nederlandse betrokkenheid op het niveau van distinctieve dienst- en productgroepen op online anonieme markten kan bepaald worden aan de hand van het Data Mining proces, waarbij er achtereenvolgend probleemdefinitie, acquisitie van achtergrondinformatie, dataselectie, pre-processing, analyse & interpretatie en rapportage en gebruik van de resultaten wordt uitgevoerd. Het gebruik van het Data Mining proces biedt een fundering waarbij de validiteit van de opeenvolging van de stappen in de ontworpen methode gewaarborgd is. Tevens biedt het een eenvoudig te volgen stappenplan, wat past bij de data die beschikbaar is voor politie- en opsporingsdiensten. Tijdens de analysestap wordt er voor het vinden van de productgroepen gebruik gemaakt van een inductieve aanpak aan de hand van clustering door het gebruik van Latent Dirichlet Allocation, waardoor de productgroepen zonder voorkennis over de categorieën in de data gevonden kunnen worden. Voor de geo-attributie wordt er gebruik gemaakt van een landspecifieke Gazetteer, bestaande uit de tien grootste steden van het land (gezien inwonersaantal) en de naam van het land zelf. Omdat de geo-attributie wordt toegepast op de onbewerkte data, is er rekening gehouden met mogelijke spel- of typefouten bij het opstellen van de woordenlijst voor de Gazetteer. Het toepassen van deze methode leidt tot een overzicht van de productgroepen per categorie, waarbij er per productgroep inzichtelijk is hoeveel listings van de productgroep een attributie naar Nederland bevatten. Naast dat deze methode een bijdrage levert aan de huidige wetenschappelijke literatuur, is de methode van toegevoegde waarde in de praktijk. De verkregen inzichten kunnen door politie- en opsporingsdiensten gebruikt worden voor het opbouwen van een intelligencepositie op het aanbod van online anonieme markten, waardoor er onderbouwde beleidskeuzes wat betreft de handhavingsstrategie van de online anonieme markten gemaakt kunnen worden. Bij toepassing van de methode op eigen opsporingsdata, zou het inzicht ook kunnen leiden tot een eenvoudige identificatie van mogelijke Nederlandse delicten of Nederlandse verdachten.

6.2. Aanbevelingen voor verder onderzoek

Aan de hand van enkele beperkingen of niet bewandelde paden binnen dit onderzoek, kunnen er aanbevelingen voor toekomstig onderzoek gedaan worden. Er worden aanbevelingen aangedragen op het gebied van de eenvoud van de implementatie van de methode, de pre-processing van de data, de toegepaste clustering en de methoe van geo-lokalisatie.

6.2.1. Eenvoud van de implementatie

Voor de methode is er gebruik gemaakt van het Data Mining proces. Hier kan over opgemerkt worden dat, hoewel de stappen conceptueel eenvoudig te volgen zijn, er zeker de nodige technische kennis op het gebied van big data en machine learning technieken nodig is voor het succesvol kunnen uitvoeren van alle fases van de methode. Het gemak van de implementatie van de methode door politie- en opsporingsdiensten zou dus onderzocht kunnen worden, zeker wanneer het de toepassing op eventuele opsporingsdata betreft.

6.2.2. Pre-processing

Omdat de pre-processing of voorbewerking van de data voor een belangrijk deel de kwaliteit van de resultaten bepaalt, kan er bij verder onderzoek nog kritisch naar deze stappen gekeken worden om de kwaliteit van de resultaten te verhogen.

Zo waren er, ondanks dat er een stap is om PGP-keys uit de listings te verwijderen, nog enkele PGP-keys in de opgeschoonde data zichtbaar. Ook bleek uit de analyse van de lengtes van de listings dat er nog outliers in de data zitten. Met een gemiddelde lengte van 141 woorden per listing (in de onbewerkte data), kunnen de 149 listings met een lengte langer dan 2000 woorden en 1 listing met een lengte van 1019066 woorden als outliers worden bestempeld. Ook het gebruiken van een package voor autocorrect (bijv. Textblob) zou een verbetering van de kwaliteit kunnen geven, doordat nu woorden zoals accout en acount nu niet als hetzelfde woord geteld worden.

Verder kan er gedacht worden aan een analyse naar het algemene woordgebruik op online anonieme markten, wat kan leiden tot een betere lijst aan stopwoorden. Het woordgebruik zou bijvoorbeeld geanalyseerd kunnen worden aan de hand van de woordfrequentie van woorden over het gehele corpus. Tijdens dit onderzoek is er namelijk per categorie een clustering uitgevoerd, waarbij de kansverdelingen van woorden en woordfrequentie dus ook slechts per categorie in kaart zijn gebracht.

6.2.3. Clustering

De clustering is uitgevoerd aan de hand van het LDA algoritme. Hoewel het LDA algoritme zeer geschikt is voor de toepassing op tekstdocumenten, hebben meerdere onderzoeken laten zien dat LDA minder goede resultaten geeft op kortere documenten, zoals bijvoorbeeld Twitter-berichten. Een uitgangspunt van LDA is namelijk dat ieder document uit meerdere topics bestaat, terwijl de aanname is dat ieder document (listing) bij slechts één productgroep behoort. In een vervolg onderzoek kan er bijvoorbeeld gekozen worden voor Bi-Term Topic Modeling, wat vaak als alternatief voor LDA bij korte teksten gebruikt wordt. Ook zou de interpretatie van gevonden productgroepen door domein-experts gevalideerd kunnen worden

6.2.4. Geo-attributie

Zoals in hoofdstuk 2 is beschreven, zijn er meerdere methodes ontworpen om aan de hand van tekstuele data geo-attributie uit te voeren. In dit onderzoek is er slechts één methode toegepast, op iedere listing afzonderlijk. Een eerste aanpak zou kunnen zijn om niet per listing, maar voor alle listings per verkoper geo-attributie uit te voeren, vanuit de aanname dat daarmee ook andere listings, waarbij er niet per sé een attribuerende term gevonden is, ook verband zouden kunnen hebben met de geolocatie. Ook zou er nagedacht kunnen worden over welk type geolocatie het meest relevant is om geo-attributie op te plegen: de handelslocatie, de verblijfslocatie of de herkomst van de verkoper? Er zou bijvoorbeeld ook geo-attributie plaats kunnen vinden door de toepassing van native-language-identification. Nagenoeg alle listings zijn in het Engels geschreven, maar door middel van probabilistische taalmodellen zou er een schatting gemaakt kunnen worden van de schrijftaal van een verkoper.

Het meest interessante onderzoek, zou het onderzoek zijn waarbij er, bijvoorbeeld door het gebruiken van listings van reeds veroordeelde handelaars op online anonieme markten, ground truth verkregen zou kunnen worden, op basis waarvan een model getraind zou kunnen worden. Dit zou een hybride model kunnen worden waarbij zowel het netwerk van een verkoper (afgeleid aan de hand van onderlinge berichten op het platform), als het taalgebruik van de verkoper en het gebruik van specifieke geolocatie attribuerende termen meegewogen wordt.

A

Appendix A

In [1]:
```
# Disable warnings for Gensim
import warnings
warnings.filterwarnings(action='ignore', category=UserWarning, module='gensim')
warnings.filterwarnings(action='ignore', category=UserWarning, module='sklearn')

# Import different packages
import numpy as np
import pandas as pd
import nltk
from nltk import word_tokenize
from nltk.stem import WordNetLemmatizer
import gensim
import gensim.corpora as corpora
from gensim.parsing.preprocessing import STOPWORDS
import sqlite3
from langdetect import detect_langs
import re
import spacy
from textblob import TextBlob

# Sklearn
from sklearn.decomposition import LatentDirichletAllocation, TruncatedSVD
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.model_selection import GridSearchCV
from pprint import pprint

# Plotting tools
import pyLDAvis
import pyLDAvis.sklearn
import pyLDAvis.gensim
import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:
```
import os
data_folder_content = os.listdir("D:/fegmiedema/Desktop")
```

In [3]:
```
error_message = "Error: db file not available, check directory"
assert "cls_digitalMerge.db" in data_folder_content, error_message
```

In [4]:
```
conn = sqlite3.connect("D:/fegmiedema/Desktop/cls_digitalMerge.db")
```

In [5]:
```
df = pd.read_sql_query("SELECT *  FROM cls_itemsMerge", conn)
```

A.1. Fase 4: Pre-processing

A.1.1. 1. Verkennende analyse van de dataset

1.1 Hoe ziet de ruwe data in de dataset er uit?

```
In [7]: df.head()
```

Out[7]:

	hash_str	category	marketplace	title	vendor	vendor_hash	total_sales	first_observed	last_observed	prediction	prediction_number	ships_to	ships_from	description	source	sales
0	00005f750be23a5f7addb026e1e2ff20	other - guide	Silk Road 2	3-5g-devils-cocktail	sweettganjababe	a3c24c810b277c1a411194ac6ab378f3	584.09	2014-07-02	2014-10-29	Digital Goods	8.0	Australia	Australia	30 day average: 4.87\n 60 day average: 4.91...	/data/projects/markets/marketplaces_epoch2/sil...	2013-11-06,0,0.0,0,0\n2013-11-13,0,0.0,0,0\n20...
1	00019696ba886889ed7612e2938f97a0	cash-out	Alphabay	RANDOM USA TRACK2 101/201 With City-zip-state	Torcarders	4154944d7c5f74f305d179db5648ee81	30.00	2017-05-15	2017-05-15	Digital Goods	8.0	Worldwide	Worldwide	RANDOM USA TRACK2 101/201 With city-zip-state\...	/data/projects/markets/epoch4/pwoah7foa6au2pul...	2015-03-18,0,0.0,0,0\n2015-03-19,0,0.0,0,0\n20...
2	0005ea294a83143fe69089e90a8fe830	other	Evolution	Jason Ferruggia - The Renegade Diet (2011)	SteroidWarehouse	305012dbcb8e1669644d6b0167eb37b6	0.96	2015-01-14	2015-01-29	Digital Goods	8.0		Worldwide		/data/projects/markets/marketplaces_epoch2/k5z...	2014-01-13,0,0.0,0,0\n2014-01-14,0,0.0,0,0\n20...
3	000720c3991dc6b45cb76d7458455721	other - guide	Evolution	(eBook)	optiman	50e0ae980df107efac03bd1ed985a2ab	1.63	2014-11-16	2015-01-13	Digital Goods	8.0		Worldwide		/data/projects/markets/marketplaces_epoch2/k5z...	2014-01-13,0,0.0,0,0\n2014-01-14,0,0.0,0,0\n20...
4	0008018df8d0305b6888062b8423c3b4	other	Silk Road 2	rc-starter-kit-milligram-scale-and-tiny-spoons...	ReconnoiterConscious	4842d0a303c20a4ec1f99b6ccc709857	162.54	2014-02-05	2014-02-12	Misc	3.0	United States	United States	Highly accurate milligram (0.001g resolution) ...	/data/projects/markets/marketplaces/silkroad6o...	2013-11-06,0,0.0,0,0\n2013-11-13,0,0.0,0,0\n20...

De tabel cls_itemMerge bestaat uit 44578 rijen en 16 kolommen: 1. **hash_str**: a hash string belonging to this unique listing (= zelfde als item_hash uit andere tabel) 2. **category**: the category according to

the classification of Van Wegberg 3. **marketplace**: the marketplace the listing was originally posted to 4. **title**: the title of the listing 5. **vendor**: the name of the vendor that posted the listing 6. **vendor_hash**: the hash string is vendor and marketplace specific 7. **total_sales**: the amount of dollars the vendor has generated by this listing. Computed from the feedback table 'feedback_items_Merge' 8. **first_observed**: the date of the first time this listing was observed in a scrape 9. **last_observed**: the date of the last time this listing was observed in a scrape 10. **prediction**: the prediction by the classification from Soska & Christin 11. **prediction_number**: the number corresponding to the prediction from Soska & Christin 12. **ships_to**: the country or continent the vendor indicated to ship the items to 13. **ships_from**: the country or contint the vendor indicated to ship the items from 14. **description**: a textual description of the good(s) or service(s) sold, written by the vendor 15. **source**: the sourcefile directory 16. **sales**: the range of the dates from when until when the listing is sold (has corresponding feedback)

1.2 Beschrijvende statistieken

```
In [8]: df.describe(include = 'all')
```

```
Out[8]:                                 hash_str  category marketplace  \
        count                              44578     44578       44578
        unique                             44578        17           8
        top     7b6e87015a0db3a383e202a7db0be2e6  cash-out    Alphabay
        freq                                   1     12324       21350
        mean                                 NaN       NaN         NaN
        std                                  NaN       NaN         NaN
        min                                  NaN       NaN         NaN
        25%                                  NaN       NaN         NaN
        50%                                  NaN       NaN         NaN
        75%                                  NaN       NaN         NaN
        max                                  NaN       NaN         NaN

                                            title vendor  \
        count                               44578  44578
        unique                              42745   5585
        top     The Essential Underground Handbook   fake
        freq                                    7   1332
        mean                                  NaN    NaN
        std                                   NaN    NaN
        min                                   NaN    NaN
        25%                                   NaN    NaN
        50%                                   NaN    NaN
        75%                                   NaN    NaN
        max                                   NaN    NaN

                                  vendor_hash    total_sales first_observed  \
        count                           44578   44578.000000          44578
        unique                           6247            NaN           1742
        top     12a2c26617922caeb288e0adb57557c6            NaN     2013-11-06
        freq                              553            NaN            495
        mean                              NaN     660.234972            NaN
        std                               NaN    5654.936751            NaN
        min                               NaN       0.000000            NaN
        25%                               NaN      13.500000            NaN
        50%                               NaN      62.510000            NaN
        75%                               NaN     280.000000            NaN
        max                               NaN  733247.520000            NaN

               last_observed     prediction  prediction_number   ships_to  ships_from  \
        count          44578          44578       44578.000000      44578       44578
```

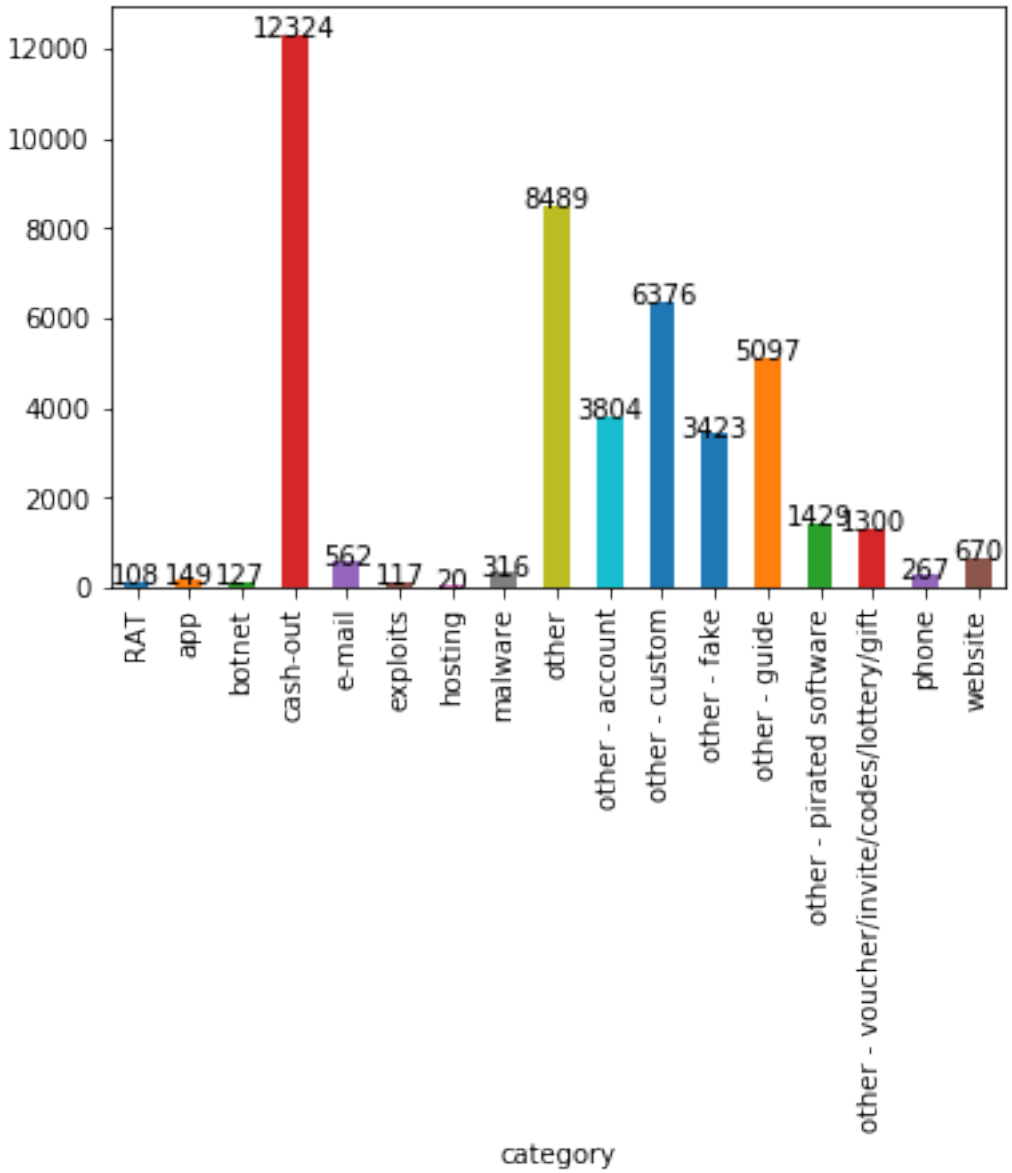
```
unique          1617              2               NaN        380         270
top       2013-11-20  Digital Goods               NaN  Worldwide   Worldwide
freq             642          36447               NaN      25592       14939
mean             NaN            NaN          7.088003        NaN         NaN
std              NaN            NaN          1.930888        NaN         NaN
min              NaN            NaN          3.000000        NaN         NaN
25%              NaN            NaN          8.000000        NaN         NaN
50%              NaN            NaN          8.000000        NaN         NaN
75%              NaN            NaN          8.000000        NaN         NaN
max              NaN            NaN          8.000000        NaN         NaN

       description                    source  \
count        44578                     44578
unique       30481                     40671
top                mysql_silkroad_1376863836
freq          9204                      1668
mean           NaN                       NaN
std            NaN                       NaN
min            NaN                       NaN
25%            NaN                       NaN
50%            NaN                       NaN
75%            NaN                       NaN
max            NaN                       NaN

                                                    sales
count                                               44578
unique                                              43883
top     2014-04-14,0,0.0,0,0\n2014-04-15,0,0.0,0,0\n20...
freq                                                   46
mean                                                  NaN
std                                                   NaN
min                                                   NaN
25%                                                   NaN
50%                                                   NaN
75%                                                   NaN
max                                                   NaN
```

The 44578 listings are from 8 marketplaces and divided in 17 categories from Van Wegberg, belonging to two type of predictions from Christin & Soska. There are 5585 different vendor names, from 6247 unique vendors (*the number of unique vendor hashes is higher than the number of unique vendor names, because vendor names are used multiple times across platforms, while the hash is platform-specific*). Those vendors used 42745 unique titles in their listings. Each listing is sold for an average 660 dollar, with the highest grossing listing earning 733.247 dollar. The goods/services are shipped from 270 different countries/continents to 380 different countries/continents (*different spelling used for countries/contintents accross platforms gives more unique hits*).

1.3 Hoeveel listings per categorie zijn er?

```
In [9]: categories = df.groupby('category')['hash_str'].nunique()

In [10]: df['category'].unique()

Out[10]: array(['other - guide', 'cash-out', 'other', 'other - account',
                'other - custom', 'other - pirated software', 'other - fake',
                'app', 'other - voucher/invite/codes/lottery/gift', 'e-mail',
                'phone', 'website', 'malware', 'exploits', 'hosting', 'RAT',
                'botnet'], dtype=object)
```

```
In [11]: ax = categories.plot(kind='bar')
         for p in ax.patches: ax.annotate(np.round(p.get_height(),decimals=0), (p.get_x()+p.get_width()
```

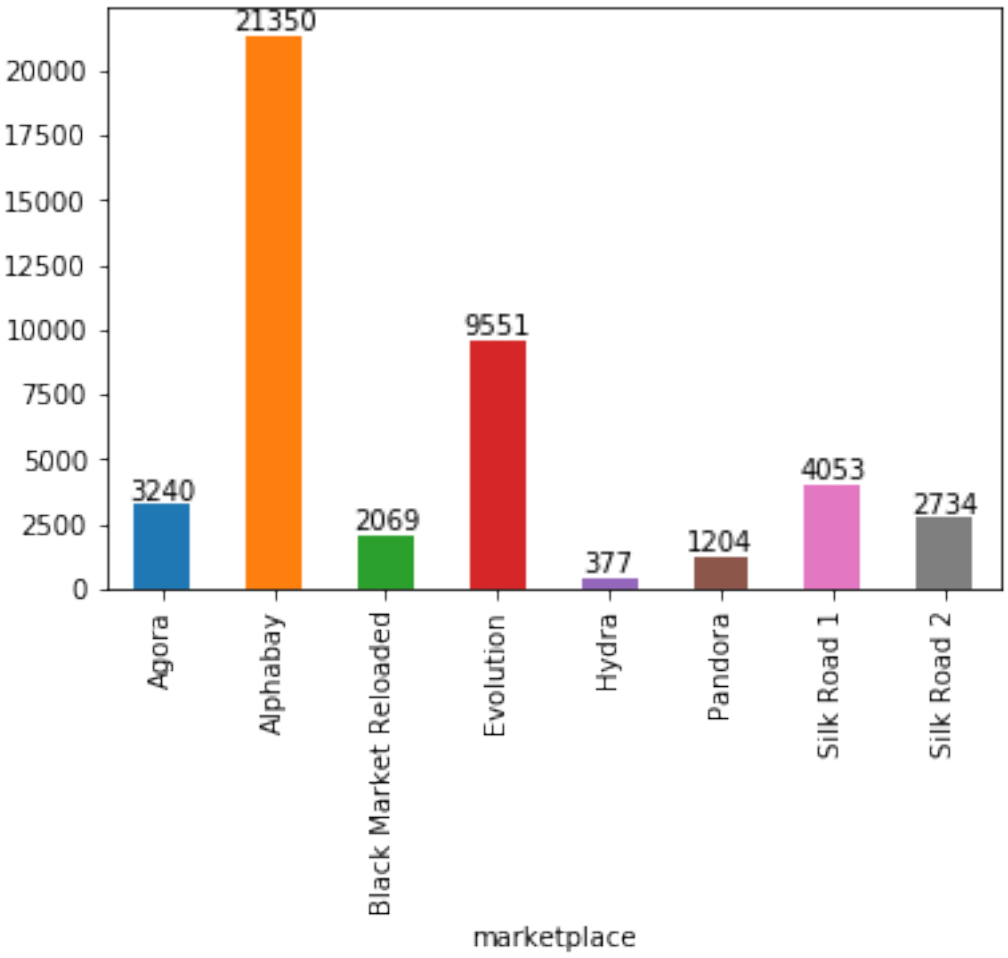


Er zijn 17 verschillende categorieën, waarbij 'cashout' (12.324 listings) en 'other' (8.489 listings) de grootste categorieën zijn. Er zijn 13 categorieën (~76%) die minder dan 4457 listings (<10% van het totaal aantal listings) in hun categorie hebben en 7 categorieën (~41%) die zelfs minder dan 445 listings (<1% van het totaal aantal listings) hebben. Oftewel: 90% van alle listings valt in de categorieën 'cash-out', 'other', 'other - custom' en 'other - guide'.

1.4 Hoe zijn de gescrapte listings verdeeld over de verschillende darkmarkets?

```
In [12]: markets = df.groupby('marketplace')['hash_str'].nunique()
```

```
In [13]: bx = markets.plot(kind='bar')
         for p in bx.patches: bx.annotate(np.round(p.get_height(),decimals=0), (p.get_x()+p.get_width()
```



Net iets minder dan de helft van alle listings (~48%) is afkomstig van de Alphabay market.

1.5 Hoeveel vendors zijn er? Hoeveel unieke vendors?

```
In [14]: df['vendor'].value_counts().shape

Out[14]: (5585,)
```

Er worden over de verschillende marktplaatsen 5585 verschillende vendornames gebruikt. De vendorname 'fake' wordt bijvoorbeeld 1332 keer gebruikt en de naam captainpicard 558 keer. Omdat de namen niet marktplaats-specifiek zijn, kan het zo zijn dat eenzelfde naam (bijv. 'DutchMasters') op drie verschillende marktplaatsen wordt gebruikt, maar hier niet dezelfde persoon achter zit. Zo worden soms 'betrouwbare namen' die gebruikt worden op een bepaald platform door andere criminelen gebruikt op een ander platform. Om er dus achter te komen hoe veel unieke verkopers er zijn, kan er beter gekeken worden naar vendor_hash: een hash van de vendor die marktplaatsspecifiek is.

```
In [15]: df['vendor_hash'].value_counts().shape

Out[15]: (6247,)
```

Hieruit blijkt dat er 6247 marktplaatsspecifieke verkopers zijn. Er zijn 130 verkopers die meer dan 50 listings op eenzelfde marktplaats hebben geplaatst:

```
In [16]: many_sales = df['vendor_hash'].value_counts()
         many_sales[many_sales > 50].count()
```

```
Out[16]: 130
```

En er zijn 2320 verkopers die slechts 1 listing hebben geplaatst:

```
In [17]: single_sale = df['vendor_hash'].value_counts()
         single_sale[single_sale < 2].count()

Out[17]: 2320
```

1.5 Inzicht: Lege velden worden niet als NaN values herkend in het dataframe

```
In [18]: df[['description','ships_to','ships_from']].head()

Out[18]:                                          description       ships_to  \
         0  30 day average: 4.87\n    60 day average: 4.91...      Australia
         1  RANDOM USA TRACK2 101/201 With city-zip-state\...      Worldwide
         2
         3
         4  Highly accurate milligram (0.001g resolution) ...  United States

               ships_from
         0      Australia
         1      Worldwide
         2      Worldwide
         3      Worldwide
         4  United States

In [19]: df[['description','ships_to','ships_from']].isnull().any()

Out[19]: description    False
         ships_to       False
         ships_from     False
         dtype: bool
```

1.5 Inzicht: Veel Silk Road 2 beschrijvingen zijn een weergave van de gemiddelde klantwaarderingen:

```
In [20]: searchfor = ['30 day average', 'Overall average: ']
         contains_average = df['description'].str.contains('|'.join(searchfor),na=False)
         df[contains_average][['title','description']][10:15]

Out[20]:                                                  title  \
         249  asus-17-3-inch-gaming-laptop-geforce-gtx-880m-...
         365                                           survive
         375  every-silk-road-s-money-making-methods-guides-...
         388                          100-x-red-apples-new-stock
         409      1-troy-ounce-gold-coin-american-eagle-bullion

                                                 description
         249  30 day average: 4.94\n    60 day average: 4.95...
         365                              Overall average: 4.99
         375  30 day average: 4.91\n    60 day average: 4.91...
         388  30 day average: 4.86\n    60 day average: 4.90...
         409  30 day average: 5.00\n    60 day average: 4.83...

In [21]: df[contains_average]['description'].count()

Out[21]: 1745
```

All deze listings zijn afkomstig van Silk Road 2:

```
In [22]: df[contains_average]['marketplace'].str.contains('Silk Road 2').count()
```

```
Out[22]: 1745
```

De data van Silk Road 2 bestaat uit 195 snapshots, en listings met dit soort beschrijvingen zijn specifiek terug te leiden naar de snapshots gemaakt na 30-08-2014:

```
In [23]: only_silk_road_2 = df['marketplace'].str.contains('Silk Road 2')
         searchfor = ['20140830', '20140902', '20140909', '20140917', '20140929', '20141003', '20
         snapshots_after_date = df[only_silk_road_2]['source'].str.contains('|'.join(searchfor),n
         df[only_silk_road_2][snapshots_after_date].shape
```

```
Out[23]: (1745, 16)
```

Ter controle: de listings gevonden via het filteren op 'day average/overall' zijn dezelfde listings als de listings gevonden via het filteren op de snapshots gemaakt na 30-08-2014. Alle listings zijn hetzelfde, dus de 'set difference' is een lege array:

```
In [23]: np.setdiff1d(df[contains_average]['hash_str'], df[only_silk_road_2][snapshots_after_date
```

```
Out[23]: array([], dtype=object)
```

1.6 Inzicht: Er zijn nog redelijk veel listings die gericht zijn op de verkoop van drugs. Dit is te zien aan het gebruik van bijv. 'mg' 'gram', of de naam van een type drugs in de title of description.

```
In [24]: searchfor = ['10g', 'MDMA', 'NaGHB', '25mg', 'ambien', 'ritalin']
         contains_drugs = df['description'].str.contains('|'.join(searchfor),na=False)
         df[contains_drugs][['title','description']][15:20]
```

```
Out[24]:                                                  title  \
         1023                                  custom for penguin
         1143                           Red Toyota - 200mg - 10 pc
         1152          *Custom listing 1 for user: princeofpeace
         1185                                 10 7.5/325mg Vicoden
         1262  IPH Isopropylphenidate {1gram} GREAT for Study...

                                                   description
         1023                              10g mdma, 100 hits lsd
         1143                           Strong MDMA pills from EU.
         1152  This custom listing contains:\r3x 30 oxycotin ...
         1185  10 7.5/325mg Vicoden\n0.12953591 BTC\n\n\r\n  ...
         1262  We ship from the USA, FAST! Yes! We Accept Int...
```

Deze listings die deze termen bevatten, zijn echter niet allemaal drugs. De listings betreffen bijv. ook materiaal om drugs mee te verwerken, zoals bijvoorbeeld erg nauwkeurige weegschalen, custom listings of er is gebruik gemaakt van STO (seach term optimalization zoals beschreven in Van Wegberg, 2018):

```
In [25]: df[contains_drugs][['title','description']][0:5]
```

```
Out[25]:                                                  title  \
         4    rc-starter-kit-milligram-scale-and-tiny-spoons...
         101   eGift Cards  Neiman Marcus - 60% OFF  Doll...
         219                         forged-social-security-card
         270                                 custom for street
         278  FULL ESCROW 10 x JUMBO the successor of tomorr...

                                                   description
         4    Highly accurate milligram (0.001g resolution) ...
         101  PayPal checkout: https://stocard.co/\r These e...
         219  This is a listing for a forged social security...
         270  200 tabs high 90s lsd\r  We aim for quick deli...
         278  !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!We Don't Ship ...
```

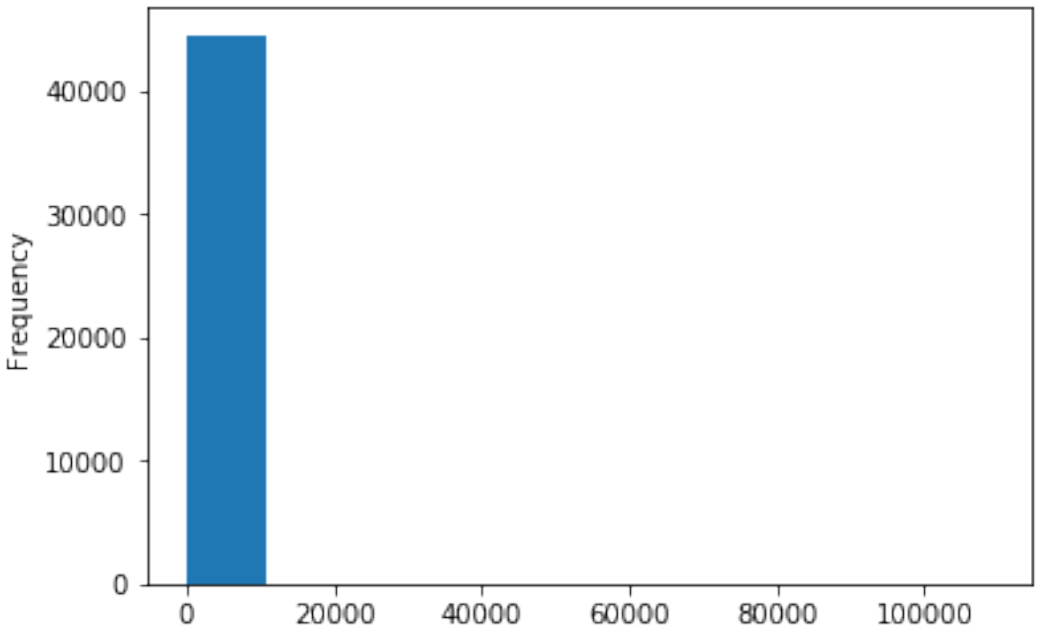
1.7 Inzicht: Het aantal woorden per listing is niet ontzettend hoog (aan de lage kant voor LDA)

```
In [26]: df['title&description'] = df['title'] + ' ' + df['description']
         df['no_of_words'] = df['title&description'].apply(lambda x: len(str(x).split(' ')))
         df.no_of_words.describe()
```

```
Out[26]: count     44578.000000
         mean        141.145588
         std         641.402770
         min           2.000000
         25%          10.000000
         50%          47.000000
         75%         148.000000
         max      109244.000000
         Name: no_of_words, dtype: float64
```

```
In [27]: df.no_of_words.plot.hist()
```

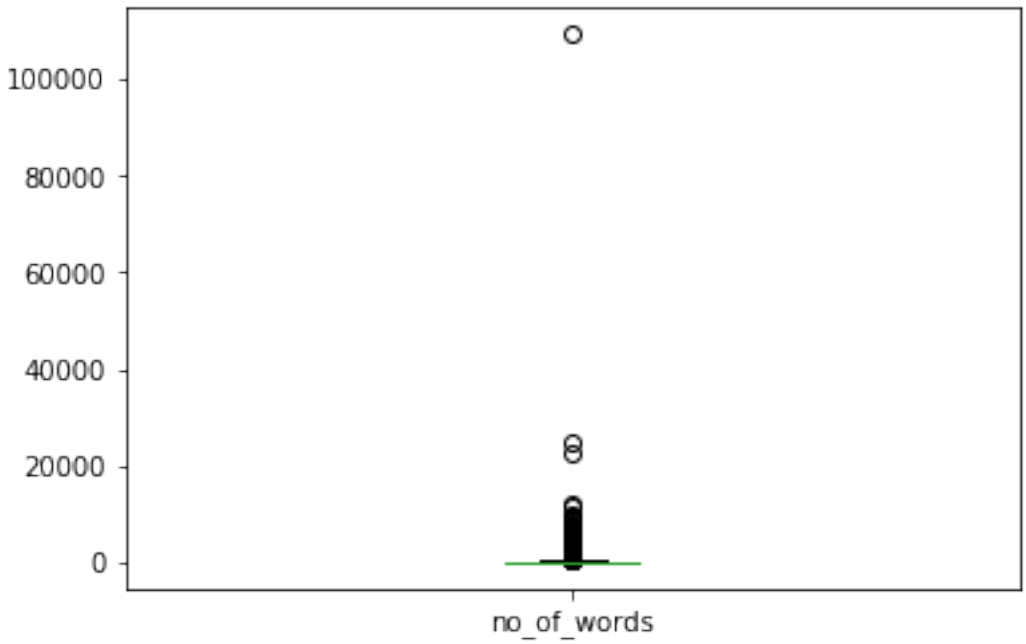
```
Out[27]: <matplotlib.axes._subplots.AxesSubplot at 0x3ff9c358>
```



Interessant om de max te bekijken. Er lijkt een redelijke outlier in de data te zitten:

```
In [28]: df.no_of_words.plot.box()
```

```
Out[28]: <matplotlib.axes._subplots.AxesSubplot at 0x3ff43080>
```



Hier lijkt inderdaad wat interessants in de hand te zijn. Verder onderzoeken hoeveel listings een lengte hebben die groter of gelijk is aan 2000 woorden:

```
In [29]: lotsofwords = df['no_of_words'] >= 2000
         df[lotsofwords]['title&description'].shape

Out[29]: (149,)
```

Misschien zijn het packages?

```
In [30]: package = df[lotsofwords]['title&description'].str.contains('package')
         df[lotsofwords][package].shape

Out[30]: (35, 18)
```

1.8 Inzicht: De listings zijn niet allemaal in het Engels geschreven
Testing whether there are different languages than English used

```
In [31]: from langdetect import DetectorFactory
         DetectorFactory.seed = 0

In [32]: language_probability = detect_langs(df['title&description'][1231:1232].to_string())
         language_probability

Out[32]: [fr:0.9999959788508584]

In [33]: def mydetect(txt):
             try:
                 return detect_langs(txt)
             except:
                 return None

In [34]: df['Language_detect'] = df['title&description'].apply(mydetect)
```

```
In [97]: languages = df['Language_detect'].notna()
         df[languages]['Language_detect'].head(15)

Out[97]: 0     [sv:0.5714265641748908, no:0.14285913661495864...
         1                                [en:0.9999977179307323]
         2       [de:0.8571389748967163, en:0.14285949375125004]
         3                                [af:0.9999932015593953]
         4                                [en:0.9999977513173884]
         5                                [en:0.9999992593967696]
         6                                [en:0.9999976369923527]
         7                                 [en:0.999995239621279]
         8                                [en:0.9999954602875811]
         9                                [en:0.9999964244843814]
         10                               [en:0.9999973335890447]
         11                               [en:0.9999982838180976]
         12                               [en:0.9999970475320276]
         13                               [fr:0.9999973579442147]
         14                               [en:0.9999957733012116]
         Name: Language_detect, dtype: object

In [106]: sorted_langs = df[languages]['Language_detect'].sort_values(ascending=True)
          sorted_langs.head(10)

Out[106]: 2489     [ro:0.28571344361958617, es:0.2857122583395601...
          4215     [en:0.285713838993789, no:0.2857135724308405, ...
          12871    [it:0.28571393641055903, hr:0.2857135284871371...
          19285    [en:0.28571411134874597, af:0.2857133939967681...
          12533    [cy:0.2857143983249749, da:0.285712793182936, ...
          17264    [pt:0.28571447403033645, es:0.2857136630266459...
          4867     [it:0.28571478996857286, sv:0.2857135696396902...
          6216     [sk:0.2857155650361808, hr:0.2857136804642817,...
          29677    [en:0.28571673817168, fr:0.285712776714993, pt...
          16669    [pt:0.29334233195660575, ro:0.2857665018745636...
          Name: Language_detect, dtype: object
```

B

Appendix B

In [1]:
```
# Import different packages
import numpy as np
import pandas as pd
import nltk
from nltk import word_tokenize
from nltk.stem import WordNetLemmatizer
import sqlite3
from langdetect import detect_langs
import re
import spacy
from textblob import TextBlob
from pprint import pprint

# Disable warnings for Gensim and SK-learn
import warnings
warnings.filterwarnings(action='ignore', category=UserWarning, module='gensim')
warnings.filterwarnings(action='ignore', category=UserWarning, module='sklearn')

# Import Gensim and SK-Learn
import gensim
import gensim.corpora as corpora
from gensim.parsing.preprocessing import STOPWORDS
from sklearn.decomposition import LatentDirichletAllocation, TruncatedSVD
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.model_selection import GridSearchCV

# Plotting tools
import pyLDAvis
import pyLDAvis.sklearn
import pyLDAvis.gensim
import matplotlib.pyplot as plt
%matplotlib inline
```

In [37]:
```
#sent_topics_sorted_cashout.to_csv('D:/fegmiedema/Desktop/cashout1.csv')
 import pandas as pd
 from pandas import ExcelWriter
 from pandas import ExcelFile
 import numpy as np
```

In [2]:
```
import os
data_folder_content = os.listdir("D:/fegmiedema/Desktop")
```

```
error_message = "Error: db file not available, check directory"
assert "cls_digitalMerge.db" in data_folder_content, error_message
conn = sqlite3.connect("D:/fegmiedema/Desktop/cls_digitalMerge.db")
df = pd.read_sql_query("SELECT *  FROM cls_itemsMerge", conn)
```

B.1. Fase 4: Pre-Processing

B.1.1. 2. Opschonen en transformeren van de data

2.1 Opschonen

```
In [3]: searchfor = ['30 day average', 'Overall average']
        df.loc[df.description.str.contains('|'.join(searchfor), na=False), 'description'] = ' '
```

2.2 Transformeren

A) Verwijderen van URLs, emails, PGP-keys, leestekens, getallen en accenten van unicode tekens, alles lowercase

```
In [4]: df['title&description'] = df['title'] + ' ' + df['description']

In [5]: def only_words(value):

            if(type(value) == type("string")):
                str1 = re.sub(r'http\S+', ' url ',value) # Delete URL's and replace them by 'url'
                str2 = re.sub(r'\S*@\S*\s?', ' email ',str1) # Delete email-addresses and replace
                str3 = re.sub(r'-----BEGIN.+-----END.+BLOCK-----', ' PGPkey ',str2, flags=re.DOTA
                str4 = re.sub(r'\\r', ' ',str3) # Delete '\\r'
                str5 = re.sub(r'\\n', ' ',str4) # Delete '\\n'
                str6 = re.sub(r'[^A-Za-z ]+', ' ',str5) # Only keep charachters (removes all punc
                str7 = re.sub(r'\b\w{,1}\b', '',str6) # Only keep words with at least two charact
                str8 = re.sub(r'\b\w{28,}\b', '',str7) # Only keep words with a maximum of 28 cha
                str9 = re.sub(r' +',' ',str8) # Remove trailing whitespaces between words
                return str9

            else:
                return np.nan

In [6]: df['title&description'] = df['title&description'].apply(only_words)
        df['title&description'] = df['title&description'].str.lower()
        df['title&description'] = df['title&description'].str.lstrip()
        df2 = df
```

B) Verwijderen van Engelse stopwoorden

```
In [7]: from nltk.stem.wordnet import WordNetLemmatizer
        from sklearn.feature_extraction import text

        my_stop_words = text.ENGLISH_STOP_WORDS.union(['john', 'david', 'please', 'positive', 'go
        stop = set(my_stop_words)

In [8]: df['title&description'] = df['title&description'].apply(lambda x: ' '.join([word for word

In [9]: df2['title&description'] = df2['title&description'].apply(lambda x: ' '.join([word for wo
```

C) Lemmatization & Tokanization

```
In [12]: warnings.simplefilter("ignore", DeprecationWarning)
         import en_core_web_sm
         nlp = en_core_web_sm.load(disable=['ner'])
         nlp.remove_pipe('parser')
         nlp.max_length = 1019066
```

```
In [11]: def lemmatization(s):
             """https://spacy.io/api/annotation"""
             allowed_postags=['NOUN', 'ADJ', 'VERB', 'ADV', 'CONJ', 'PROPN', 'X']
             texts_out = []
             for i in s:
                 doc = nlp(i)
                 texts_out.append([token.lemma_ for token in doc if (token.pos_ in allowed_postags) and
             return texts_out
```

D) Opdelen in categorieën voor analyse

```
In [13]: cashout = df2['category'].str.contains('cash-out')
         app = df2['category'].str.contains('app')
         RAT = df2['category'].str.contains('RAT')
         botnet = df2['category'].str.contains('botnet')
         email = df2['category'].str.contains('e-mail')
         exploits = df2['category'].str.contains('exploits')
         hosting = df2['category'].str.contains('hosting')
         malware = df2['category'].str.contains('malware')
         other = df2['category'].str.contains('other')
         other_account = df2['category'].str.contains('account')
         other_custom = df2['category'].str.contains('custom')
         other_fake = df2['category'].str.contains('fake')
         other_guide = df2['category'].str.contains('guide')
         other_pirated = df2['category'].str.contains('pirated')
         other_voucher = df2['category'].str.contains('voucher')
         phone = df2['category'].str.contains('phone')
         website = df2['category'].str.contains('website')

In [14]: cashout_texts = df2[cashout]['title'] + ' ' + df[cashout]['description']
         app_texts = df2[app]['title'] + ' ' + df[app]['description']
         RAT_texts = df2[RAT]['title'] + ' ' + df[RAT]['description']
         botnet_texts = df2[botnet]['title'] + ' ' + df[botnet]['description']
         email_texts = df2[email]['title'] + ' ' + df[email]['description']
         exploits_texts = df2[exploits]['title'] + ' ' + df[exploits]['description']
         hosting_texts = df2[hosting]['title'] + ' ' + df[hosting]['description']
         malware_texts = df2[malware]['title'] + ' ' + df[malware]['description']
         other_texts = df2[other]['title'] + ' ' + df[other]['description']
         other_account_texts = df2[other_account]['title'] + ' ' + df[other_account]['description']
         other_custom_texts = df2[other_custom]['title'] + ' ' + df[other_custom]['description']
         other_fake_texts = df2[other_fake]['title'] + ' ' + df[other_fake]['description']
         other_guide_texts = df2[other_guide]['title'] + ' ' + df[other_guide]['description']
         other_pirated_texts = df2[other_pirated]['title'] + ' ' + df[other_pirated]['description']
         other_voucher_texts = df2[other_voucher]['title'] + ' ' + df[other_voucher]['description']
         phone_texts = df2[phone]['title'] + ' ' + df[phone]['description']
         website_texts = df2[website]['title'] + ' ' + df[website]['description']

In [15]: from gensim.models import CoherenceModel

In [16]: def compute_coherence_values(dictionary, corpus, texts, limit, start, step):
             coherence_values = []
             model_list = []
             for num_topics in range(start, limit, step):
                 model = gensim.models.ldamodel.LdaModel(corpus=corpus,
                                                         id2word=dictionary,
                                                         num_topics=num_topics,
                                                         random_state=100,
```

```
                                                update_every=1,
                                                chunksize=100,
                                                passes=10,
                                                alpha='auto',
                                                per_word_topics=True)
            model_list.append(model)
            coherencemodel = CoherenceModel(model=model, texts=texts, dictionary=dictionary,
            coherence_values.append(coherencemodel.get_coherence())

    return model_list, coherence_values
```

B.2. Fase 5: Analyse

Geo-attributie

```
In [188]: searchfor = ['nederlands', 'Nederlands', 'Nederland', 'nederland', 'Netherlands', 'neth
          df2['dutch_title'] = df2['title'].str.contains('|'.join(searchfor),na=False)
          df2['dutch_vendor'] = df2['vendor'].str.contains('|'.join(searchfor),na=False)
          df2['dutch_description'] = df2['description'].str.contains('|'.join(searchfor),na=False
```

```
In [189]: df2['dutch'] = (df2['dutch_vendor']) | (df2['dutch_title'] | (df2['dutch_description'])
```

```
In [190]: df2[df2['dutch']].shape
```

```
Out[190]: (1172, 21)
```

B.2.1. Cashout - 12324 listings - 11 clusters

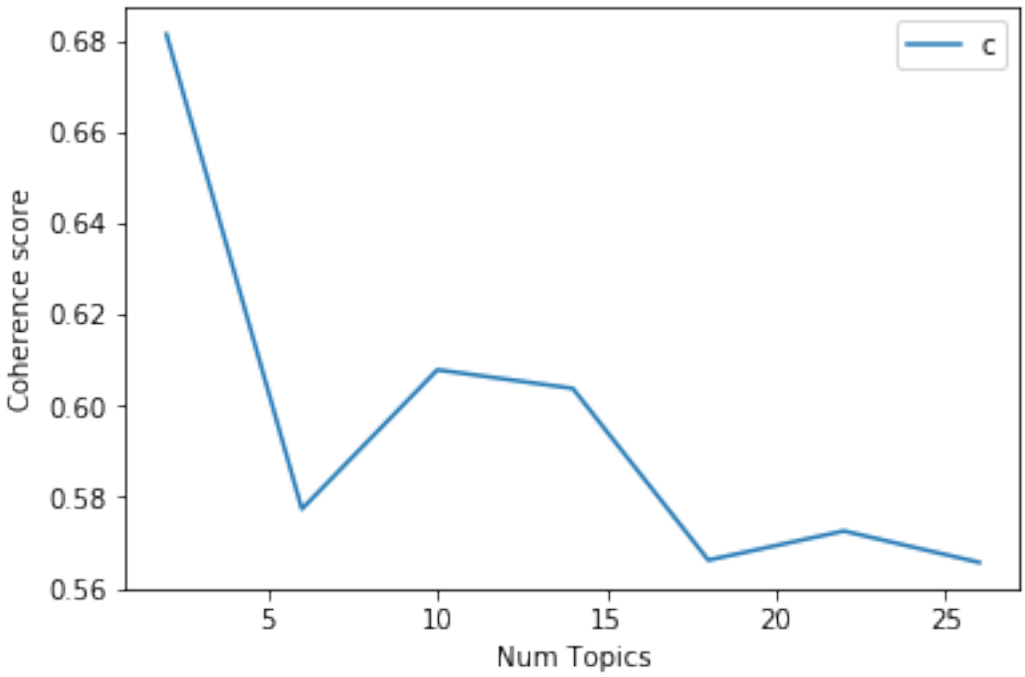
```
In [17]: data_lemmatized_cashout = lemmatization(df2[cashout]['title&description'])
```

```
In [18]: id2word_cashout = corpora.Dictionary(data_lemmatized_cashout)
         texts_cashout = data_lemmatized_cashout
         corpus_cashout = [id2word_cashout.doc2bow(text) for text in texts_cashout]
```

Run het model op 2 t/m 30 clusters, met stappen van 4

```
In [39]: # Can take a long time to run.
         model_list_cashout, coherence_values_cashout = compute_coherence_values(dictionary=id2wo
```

```
In [40]: # Show graph
         limit=30; start=2; step=4;
         x = range(start, limit, step)
         plt.plot(x, coherence_values_cashout)
         plt.xlabel("Num Topics")
         plt.ylabel("Coherence score")
         plt.legend(("coherence_values"), loc='best')
         plt.show()
```



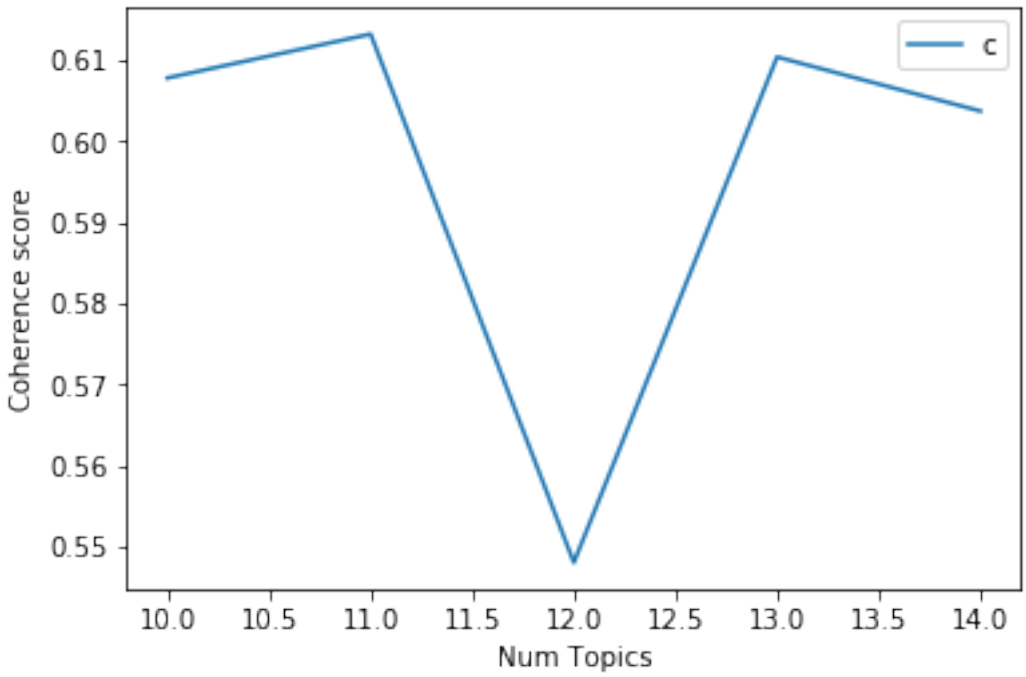

```
In [41]: # Print the coherence scores
         for m, cv in zip(x, coherence_values_cashout):
             print("Num Topics =", m, " has Coherence Value of", round(cv, 4))

Num Topics = 2  has Coherence Value of 0.6813
Num Topics = 6  has Coherence Value of 0.5774
Num Topics = 10  has Coherence Value of 0.6078
Num Topics = 14  has Coherence Value of 0.6038
Num Topics = 18  has Coherence Value of 0.5662
Num Topics = 22  has Coherence Value of 0.5725
Num Topics = 26  has Coherence Value of 0.5657
```

```
In [42]: optimal_model_cashout = model_list_cashout[2]
         vis = pyLDAvis.gensim.prepare(optimal_model_cashout, corpus_cashout, id2word_cashout)
         pyLDAvis.display(vis)

Out[42]: <IPython.core.display.HTML object>
```

Run het model op 10 t/m 15 clusters, met stappen van 1

```
In [34]: # Can take a long time to run.
         model_list_cashout2, coherence_values_cashout2 = compute_coherence_values(dictionary=id2word_c
```

```
In [35]: # Show graph
         limit=15; start=10; step=1;
         x = range(start, limit, step)
         plt.plot(x, coherence_values_cashout2)
         plt.xlabel("Num Topics")
         plt.ylabel("Coherence score")
         plt.legend(("coherence_values"), loc='best')
         plt.show()
```




```
In [36]: # Print the coherence scores
         for m, cv in zip(x, coherence_values_cashout2):
             print("Num Topics =", m, " has Coherence Value of", round(cv, 4))

Num Topics = 10  has Coherence Value of 0.6078
Num Topics = 11  has Coherence Value of 0.6132
Num Topics = 12  has Coherence Value of 0.5481
Num Topics = 13  has Coherence Value of 0.6104
Num Topics = 14  has Coherence Value of 0.6038


In [63]: optimal_model_cashout = model_list_cashout2[1]
         vis = pyLDAvis.gensim.prepare(optimal_model_cashout, corpus_cashout, id2word_cashout)
         pyLDAvis.display(vis)

Out[63]: <IPython.core.display.HTML object>

In [64]: def format_topics_sentences(ldamodel, corpus, texts, original, vendor, sales, dutch):
             # Array of top 10 topics
             top10array = []

             for row in range(ldamodel.num_topics):
                 wp = ldamodel.show_topic(row)
                 topic_keywords = ", ".join([word for word, prop in wp])
                 top10array.append((row, topic_keywords))

             top10dict = dict(top10array)

             sent_topics_df = pd.DataFrame(pd.DataFrame([sorted(topic[0], key=lambda x: (x[1]), r
             sent_topics_df.columns=["Data"]
```

```
        original = original.reset_index()
        sent_topics_df = pd.concat([sent_topics_df, original], axis=1)
        sent_topics_df.columns.values[2] = 'Original'

        vendors = vendor.reset_index()
        sent_topics_df = pd.concat([sent_topics_df, vendors], axis=1)
        sent_topics_df.columns.values[4] = 'Vendors'

        total_sales = sales.reset_index()
        sent_topics_df = pd.concat([sent_topics_df, total_sales], axis=1)
        sent_topics_df.columns.values[6] = 'Total_sales'

        dutch_id = dutch.reset_index()
        sent_topics_df = pd.concat([sent_topics_df, dutch_id], axis=1)
        sent_topics_df.columns.values[8] = 'Dutch_id'

        sent_topics_df['Original'] = sent_topics_df.Original
        sent_topics_df['Vendors'] = sent_topics_df.Vendors
        sent_topics_df['Total_sales'] = sent_topics_df.Total_sales
        sent_topics_df['Dutch_id'] = sent_topics_df.Dutch_id
        sent_topics_df['Index'] = sent_topics_df.index
        sent_topics_df['Dominant_Topic'] = sent_topics_df.Data.apply(lambda x: x[0])
        sent_topics_df['Perc_Contribution'] = sent_topics_df.Data.apply(lambda x: round(x[1],4))
        sent_topics_df['Topic_Keywords'] = sent_topics_df.Dominant_Topic.apply(lambda x: top10dict

        sent_topics_df = sent_topics_df[['Dominant_Topic', 'Perc_Contribution', 'Topic_Keywords',
        return(sent_topics_df)

    vendor = df2[cashout]['vendor']
    sales = df2[cashout]['total_sales']
    dutch = df[cashout]['dutch']
    df2_topic_sents_keywords_cashout = format_topics_sentences(ldamodel=optimal_model_cashout, cor

    # Format
    df2_dominant_topic_cashout = df2_topic_sents_keywords_cashout.reset_index()
    df2_dominant_topic_cashout.columns = ['Document_No', 'Dominant_Topic', 'Topic_Perc_Contrib', '
```

In [65]:
```
    # Group top 5 sentences under each topic
    sent_topics_sorted_cashout = pd.DataFrame()

    sent_topics_outdf_grpd = df2_topic_sents_keywords_cashout.groupby('Dominant_Topic')

    for i, grp in sent_topics_outdf_grpd:
        sent_topics_sorted_cashout = pd.concat([sent_topics_sorted_cashout,
                                                grp.sort_values(['Perc_Contribution'], ascending=
                                               axis=0)

    # Reset Index
    sent_topics_sorted_cashout.reset_index(drop=True, inplace=True)

    # Format
    sent_topics_sorted_cashout.columns = ['Topic_Num', "Topic_Perc_Contrib", "Keywords", "Text", '
```

In [66]:
```
    cluster0 = df2_dominant_topic_cashout.loc[df2_dominant_topic_cashout['Dominant_Topic'] == 0]
    cluster1 = df2_dominant_topic_cashout.loc[df2_dominant_topic_cashout['Dominant_Topic'] == 1]
    cluster2 = df2_dominant_topic_cashout.loc[df2_dominant_topic_cashout['Dominant_Topic'] == 2]
    cluster3 = df2_dominant_topic_cashout.loc[df2_dominant_topic_cashout['Dominant_Topic'] == 3]
```

```
cluster4 = df2_dominant_topic_cashout.loc[df2_dominant_topic_cashout['Dominant_Topic'] =
cluster5 = df2_dominant_topic_cashout.loc[df2_dominant_topic_cashout['Dominant_Topic'] =
cluster6 = df2_dominant_topic_cashout.loc[df2_dominant_topic_cashout['Dominant_Topic'] =
cluster7 = df2_dominant_topic_cashout.loc[df2_dominant_topic_cashout['Dominant_Topic'] =
cluster8 = df2_dominant_topic_cashout.loc[df2_dominant_topic_cashout['Dominant_Topic'] =
cluster9 = df2_dominant_topic_cashout.loc[df2_dominant_topic_cashout['Dominant_Topic'] =
cluster10 = df2_dominant_topic_cashout.loc[df2_dominant_topic_cashout['Dominant_Topic']

d = {'Cluster_No' : pd.Series([0,1,2,3,4,5,6,7,8,9,10]),
     'Total_listings' : pd.Series([len(df2_dominant_topic_cashout)]),
     'Numb_of_list ' : pd.Series([len(cluster0), len(cluster1), len(cluster2), len(clust
     'Total_uniq_vendors' : pd.Series([df2_dominant_topic_cashout.Vendors.nunique()]),
     'Unique_vendors' : pd.Series([cluster0.Vendors.nunique(),cluster1.Vendors.nunique()
     'Total_revenue' : pd.Series([df2_dominant_topic_cashout.Total_sales.sum()]),
     'Revenue' : pd.Series([cluster0.Total_sales.sum(),cluster1.Total_sales.sum(),cluste
    'Dutch_identifier' : pd.Series([cluster0.Dutch_id.sum(),cluster1.Dutch_id.sum(),clus
df_intel = pd.DataFrame(d)
cols = df_intel.columns.tolist()
cols = ['Cluster_No',
 'Total_listings',
 'Numb_of_list ',
 'Total_uniq_vendors',
 'Unique_vendors',
 'Total_revenue',
 'Revenue',
 'Dutch_identifier']
df_intel = df_intel[cols]
```

Voorbeeld van Nederlandse geo-attributie in een cluster

```
In [73]: cluster6.loc[cluster6.Dutch_id == True, ['Original','Vendors']]
```

Out[73]:

	Original	Vendors
95	1000 X MASTERCARD 200MGS * Red Mastercard\r* D...	louboutinUK
700	[US] Verified Paypal Accounts, with CC Attache...	dutchsnowflake
732	[EU] Hacked Ebay accounts with Paypal & E-mail...	dutchsnowflake
784	10% OFF[UK] Hacked Verified Paypal Accounts wi...	dutchsnowflake
843	Worldwide Windows RDP/VPS Service Worldwide Ha...	kimera
1079	cashout wells fargo to bitcoin guide THIS TUTO...	dutchdream
1255	BULK DISCOUNT!! [UK] Hacked Verified Paypal Ac...	dutchsnowflake
1368	[US] Verified Paypal Accounts, Bank/CC Attache...	dutchsnowflake
1608	NEW!! ALL ACC. HAVE TRANSACTION IN MAY 2016!! ...	dutchsnowflake
1620	[US] Hacked Ebay accounts with Paypal & E-mail...	dutchsnowflake
1643	BUY 10, GET 10% OFF!! [US] Hacked Verified Pay...	dutchsnowflake
1697	[NL] Verified Paypal Accounts, Bank/CC Attache...	dutchsnowflake
2086	BUY 10, GET 10% OFF!! [UK] Hacked Verified Pay...	dutchsnowflake
2118	PayPal Middleman Account with Email access + S...	CashTrade
2224	[UK] Hacked Amazon accounts with CC Attached! ...	dutchsnowflake
3654	[UK] Hacked Ebay accounts with Paypal & E-mail...	dutchsnowflake
4824	NOW WITH RECENT TRANSACTION(S)!![US] Hacked Ve...	dutchsnowflake
5010	[US] Hacked Amazon accounts with CC Attached! ...	dutchsnowflake
5388	Kane's [AGED] NL Bank Account's & Login & P...	kane18
5470	[US] Hacked Verified Paypal Accounts, with CC ...	dutchsnowflake
6804	[CA] Verified Paypal Accounts, Bank/CC Attache...	dutchsnowflake
6824	[UK] Verified Paypal Accounts, Bank/CC Attache...	dutchsnowflake
6970	[EU] Verified Paypal Accounts, Bank/CC Attache...	dutchsnowflake
7030	[WORLDWIDE] Verified Paypal Accounts, Bank/CC ...	dutchsnowflake

```
7072   PAYPAL MIDDLEMAN CUSTOMIZED+ID PROOF - Cheap &...  FakeID4WorldWide
7138   [CA] Hacked Verified Paypal Accounts, Bank/CC ...    dutchsnowflake
7305                              Dutch ID card Scans HQ             Mr2011
7621   [UK] Hacked Verified Paypal Accounts with BANK...    dutchsnowflake
8174   CONvenient NL IBAN Bankdrop Method This method...       CONvenience
8199   [UK] Verified Paypal Accounts with BANK or CC ...    dutchsnowflake
10392  BULK DISCOUNTS!![US] Hacked Verified Paypal Ac...    dutchsnowflake
11221  US ONLINE BANK ACCOUNT CREATION GUIDE + FREE D...        dutchdream
11500  [FR] Verified Paypal Accounts, Bank/CC Attache...    dutchsnowflake
12252  RECENT TRANSACTION(S) 2017!![US] Hacked Verifi...    dutchsnowflake
```

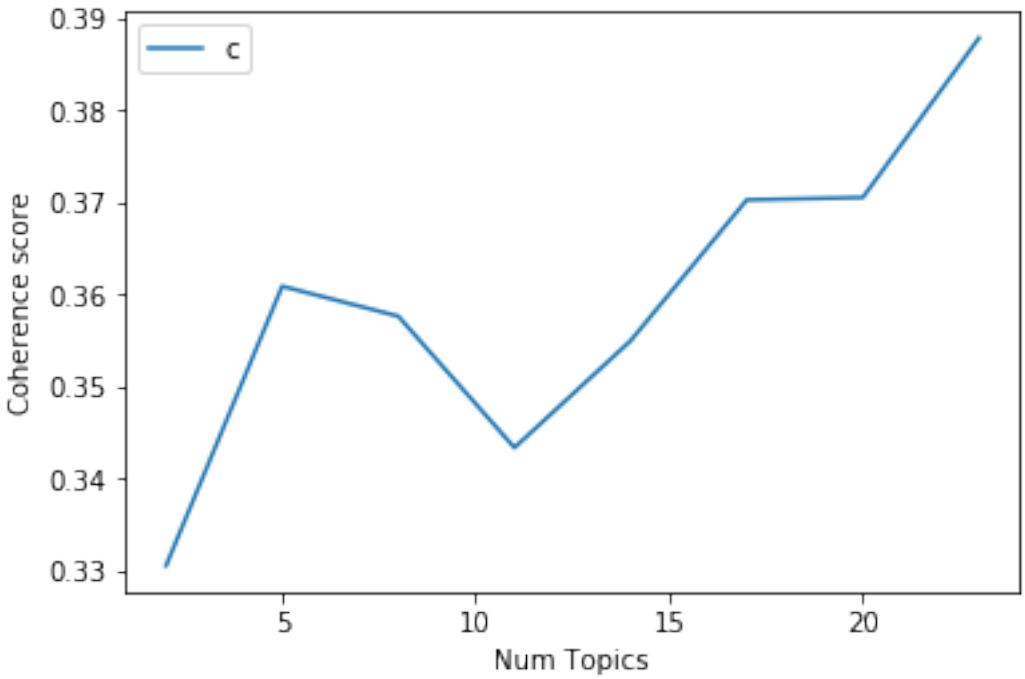
```
In [67]: writer = ExcelWriter('D:/fegmiedema/Desktop/Cashout11FINAL.xlsx')
         sent_topics_sorted_cashout.to_excel(writer,'Sheet1',index=False)
         df_intel.to_excel(writer,'Sheet2',index=False)
         writer.save()
```

B.2.2. App - 144 listings - 5 topics

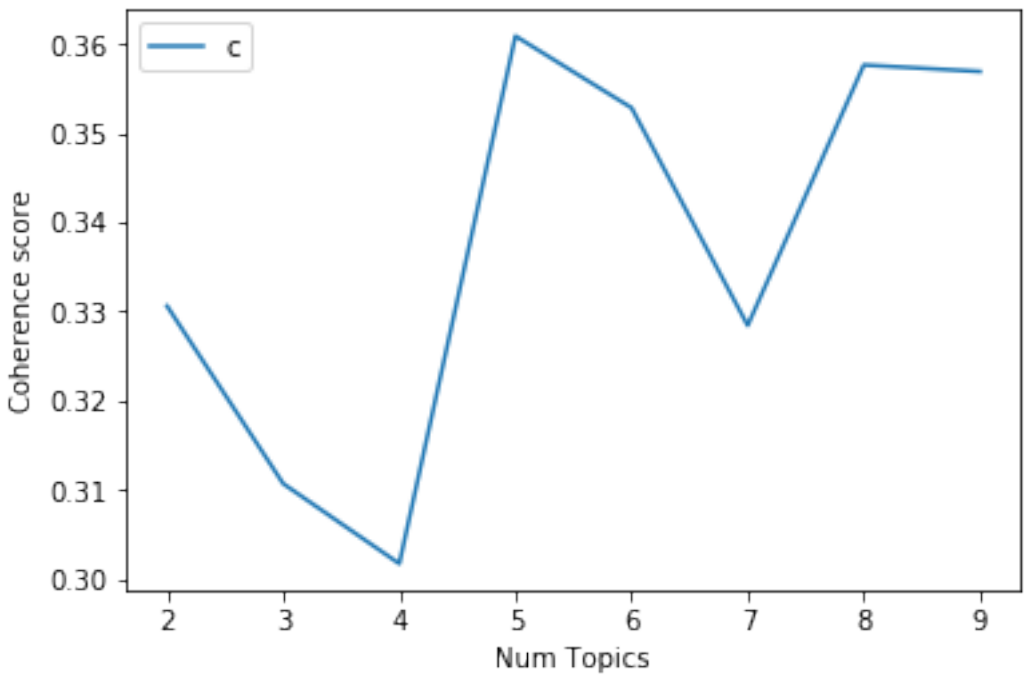
```
In [19]: data_lemmatized_app = lemmatization(df2[app]['title&description'])
         id2word_app = corpora.Dictionary(data_lemmatized_app)
         texts_app = data_lemmatized_app
         corpus_app = [id2word_app.doc2bow(text) for text in texts_app]
```

```
In [92]: model_list_app, coherence_values_app = compute_coherence_values(dictionary=id2word_app, corpus
```

```
In [93]: # Show graph
         limit=25; start=2; step=3;
         x = range(start, limit, step)
         plt.plot(x, coherence_values_app)
         plt.xlabel("Num Topics")
         plt.ylabel("Coherence score")
         plt.legend(("coherence_values"), loc='best')
         plt.show()
```



```
In [94]: # Print the coherence scores
         for m, cv in zip(x, coherence_values_app):
             print("Num Topics =", m, " has Coherence Value of", round(cv, 4))

Num Topics = 2  has Coherence Value of 0.3306
Num Topics = 5  has Coherence Value of 0.3609
Num Topics = 8  has Coherence Value of 0.3577
Num Topics = 11  has Coherence Value of 0.3434
Num Topics = 14  has Coherence Value of 0.355
Num Topics = 17  has Coherence Value of 0.3703
Num Topics = 20  has Coherence Value of 0.3705
Num Topics = 23  has Coherence Value of 0.3878


In [95]: optimal_model_app = model_list_app[1]
         vis = pyLDAvis.gensim.prepare(optimal_model_app, corpus_app, id2word_app)
         pyLDAvis.display(vis)

Out[95]: <IPython.core.display.HTML object>

In [100]: model_list_app2, coherence_values_app2 = compute_coherence_values(dictionary=id2word_ap

In [101]: # Show graph
          limit=10; start=2; step=1;
          x = range(start, limit, step)
          plt.plot(x, coherence_values_app2)
          plt.xlabel("Num Topics")
          plt.ylabel("Coherence score")
          plt.legend(("coherence_values"), loc='best')
          plt.show()
```



```
In [102]: # Print the coherence scores
          for m, cv in zip(x, coherence_values_app2):
              print("Num Topics =", m, " has Coherence Value of", round(cv, 4))
```

```
Num Topics = 2  has Coherence Value of 0.3306
Num Topics = 3  has Coherence Value of 0.3107
Num Topics = 4  has Coherence Value of 0.3017
Num Topics = 5  has Coherence Value of 0.3609
Num Topics = 6  has Coherence Value of 0.3528
Num Topics = 7  has Coherence Value of 0.3284
Num Topics = 8  has Coherence Value of 0.3577
Num Topics = 9  has Coherence Value of 0.3569
```

```
In [96]: def format_topics_sentences(ldamodel, corpus, texts, original, vendor, sales):
             # Array of top 10 topics
             top10array = []

             for row in range(ldamodel.num_topics):
                 wp = ldamodel.show_topic(row)
                 topic_keywords = ", ".join([word for word, prop in wp])
                 top10array.append((row, topic_keywords))

             top10dict = dict(top10array)

             sent_topics_df = pd.DataFrame(pd.DataFrame([sorted(topic[0], key=lambda x: (x[1]), reverse
             sent_topics_df.columns=["Data"]

             original = original.reset_index()
             sent_topics_df = pd.concat([sent_topics_df, original], axis=1)
             sent_topics_df.columns.values[2] = 'Original'

             vendors = vendor.reset_index()
             sent_topics_df = pd.concat([sent_topics_df, vendors], axis=1)
             sent_topics_df.columns.values[4] = 'Vendors'

             total_sales = sales.reset_index()
             sent_topics_df = pd.concat([sent_topics_df, total_sales], axis=1)
             sent_topics_df.columns.values[5] = 'Total_sales'

             sent_topics_df['Original'] = sent_topics_df.Original
             sent_topics_df['Vendors'] = sent_topics_df.Vendors
             sent_topics_df['Total_sales'] = sent_topics_df.Total_sales
             sent_topics_df['Index'] = sent_topics_df.index
             sent_topics_df['Dominant_Topic'] = sent_topics_df.Data.apply(lambda x: x[0])
             sent_topics_df['Perc_Contribution'] = sent_topics_df.Data.apply(lambda x: round(x[1],4))
             sent_topics_df['Topic_Keywords'] = sent_topics_df.Dominant_Topic.apply(lambda x: top10dict

             sent_topics_df = sent_topics_df[['Dominant_Topic', 'Perc_Contribution', 'Topic_Keywords',
             return(sent_topics_df)

         vendor = df2[app]['vendor']
         sales = df2[app]['total_sales']
         df2_topic_sents_keywords_app = format_topics_sentences(ldamodel=optimal_model_app, corpus=corp

         # Format
         df2_dominant_topic_app = df2_topic_sents_keywords_app.reset_index()
```

```
df2_dominant_topic_app.columns = ['Document_No', 'Dominant_Topic', 'Topic_Perc_Contrib',
```

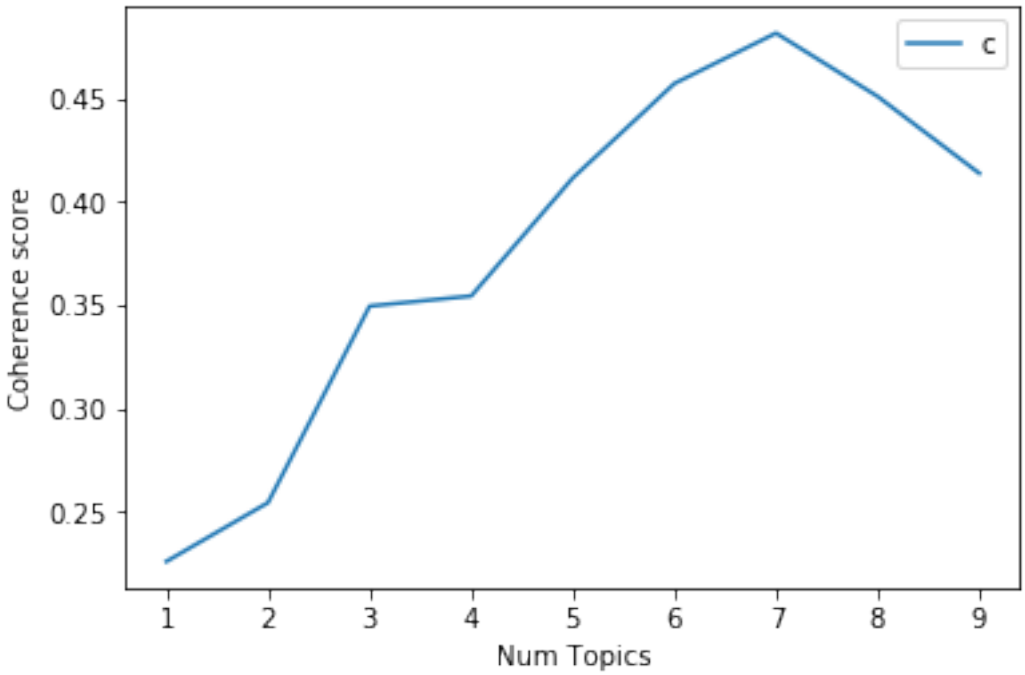
In [97]:
```
# Group top 5 sentences under each topic
sent_topics_sorted_app = pd.DataFrame()

sent_topics_outdf_grpd = df2_topic_sents_keywords_app.groupby('Dominant_Topic')

for i, grp in sent_topics_outdf_grpd:
    sent_topics_sorted_app = pd.concat([sent_topics_sorted_app,
                                        grp.sort_values(['Perc_Contribution'], asce
                                       axis=0)

# Reset Index
sent_topics_sorted_app.reset_index(drop=True, inplace=True)

# Format
sent_topics_sorted_app.columns = ['Topic_Num', "Topic_Perc_Contrib", "Keywords", "Text",
```

In [98]:
```
cluster0 = df2_dominant_topic_app.loc[df2_dominant_topic_app['Dominant_Topic'] == 0]
cluster1 = df2_dominant_topic_app.loc[df2_dominant_topic_app['Dominant_Topic'] == 1]
cluster2 = df2_dominant_topic_app.loc[df2_dominant_topic_app['Dominant_Topic'] == 2]
cluster3 = df2_dominant_topic_app.loc[df2_dominant_topic_app['Dominant_Topic'] == 3]
cluster4 = df2_dominant_topic_app.loc[df2_dominant_topic_app['Dominant_Topic'] == 4]
cluster5 = df2_dominant_topic_app.loc[df2_dominant_topic_app['Dominant_Topic'] == 5]
cluster6 = df2_dominant_topic_app.loc[df2_dominant_topic_app['Dominant_Topic'] == 6]
cluster7 = df2_dominant_topic_app.loc[df2_dominant_topic_app['Dominant_Topic'] == 7]
cluster8 = df2_dominant_topic_app.loc[df2_dominant_topic_app['Dominant_Topic'] == 8]
cluster9 = df2_dominant_topic_app.loc[df2_dominant_topic_app['Dominant_Topic'] == 9]
cluster10 = df2_dominant_topic_app.loc[df2_dominant_topic_app['Dominant_Topic'] == 10]
cluster11 = df2_dominant_topic_app.loc[df2_dominant_topic_app['Dominant_Topic'] == 11]
cluster12 = df2_dominant_topic_app.loc[df2_dominant_topic_app['Dominant_Topic'] == 12]
cluster13 = df2_dominant_topic_app.loc[df2_dominant_topic_app['Dominant_Topic'] == 13]

d = {'Cluster_No' : pd.Series([0,1,2,3,4,5,6,7,8,9,10,11,12,13]),
     'Total_listings' : pd.Series([len(df2_dominant_topic_app)]),
     'Numb_of_list ' : pd.Series([len(cluster0), len(cluster1), len(cluster2), len(clust
     'Total_uniq_vendors' : pd.Series([df2_dominant_topic_app.Vendors.nunique()]),
     'Unique_vendors' : pd.Series([cluster0.Vendors.nunique(),cluster1.Vendors.nunique()
     'Total_revenue' : pd.Series([df2_dominant_topic_app.Total_sales.sum()]),
     'Revenue' : pd.Series([cluster0.Total_sales.sum(),cluster1.Total_sales.sum(),cluste
df_intel = pd.DataFrame(d)
cols = df_intel.columns.tolist()
cols = ['Cluster_No',
 'Total_listings',
 'Numb_of_list ',
 'Total_uniq_vendors',
 'Unique_vendors',
 'Total_revenue',
 'Revenue']
df_intel = df_intel[cols]
```

In [99]:
```
writer = ExcelWriter('D:/fegmiedema/Desktop/App.xlsx')
sent_topics_sorted_app.to_excel(writer,'Sheet1',index=False)
df_intel.to_excel(writer,'Sheet2',index=False)
writer.save()
```

B.2.3. RAT - 105 listings - 7 topics

```
In [20]: data_lemmatized_RAT = lemmatization(df2[RAT]['title&description'])
         id2word_RAT = corpora.Dictionary(data_lemmatized_RAT)
         texts_RAT = data_lemmatized_RAT
         corpus_RAT = [id2word_RAT.doc2bow(text) for text in texts_RAT]

In [29]: model_list_RAT, coherence_values_RAT = compute_coherence_values(dictionary=id2word_RAT, corpus

In [30]: # Show graph
         limit=10; start=1; step=1;
         x = range(start, limit, step)
         plt.plot(x, coherence_values_RAT)
         plt.xlabel("Num Topics")
         plt.ylabel("Coherence score")
         plt.legend(("coherence_values"), loc='best')
         plt.show()
```




```
In [31]: # Print the coherence scores
         for m, cv in zip(x, coherence_values_RAT):
             print("Num Topics =", m, " has Coherence Value of", round(cv, 4))
```

```
Num Topics = 1  has Coherence Value of 0.2258
Num Topics = 2  has Coherence Value of 0.2543
Num Topics = 3  has Coherence Value of 0.3496
Num Topics = 4  has Coherence Value of 0.3545
Num Topics = 5  has Coherence Value of 0.4117
Num Topics = 6  has Coherence Value of 0.4575
Num Topics = 7  has Coherence Value of 0.4819
Num Topics = 8  has Coherence Value of 0.4512
Num Topics = 9  has Coherence Value of 0.4139
```

```
In [32]: optimal_model_RAT = model_list_RAT[6]
         vis = pyLDAvis.gensim.prepare(optimal_model_RAT, corpus_RAT, id2word_RAT)
         pyLDAvis.display(vis)

Out[32]: <IPython.core.display.HTML object>

In [33]: def format_topics_sentences(ldamodel, corpus, texts, original, vendor, sales):
             # Array of top 10 topics
             top10array = []

             for row in range(ldamodel.num_topics):
                 wp = ldamodel.show_topic(row)
                 topic_keywords = ", ".join([word for word, prop in wp])
                 top10array.append((row, topic_keywords))

             top10dict = dict(top10array)

             sent_topics_df = pd.DataFrame(pd.DataFrame([sorted(topic[0], key=lambda x: (x[1]), r
             sent_topics_df.columns=["Data"]

             original = original.reset_index()
             sent_topics_df = pd.concat([sent_topics_df, original], axis=1)
             sent_topics_df.columns.values[2] = 'Original'

             vendors = vendor.reset_index()
             sent_topics_df = pd.concat([sent_topics_df, vendors], axis=1)
             sent_topics_df.columns.values[4] = 'Vendors'

             total_sales = sales.reset_index()
             sent_topics_df = pd.concat([sent_topics_df, total_sales], axis=1)
             sent_topics_df.columns.values[5] = 'Total_sales'

             sent_topics_df['Original'] = sent_topics_df.Original
             sent_topics_df['Vendors'] = sent_topics_df.Vendors
             sent_topics_df['Total_sales'] = sent_topics_df.Total_sales
             sent_topics_df['Index'] = sent_topics_df.index
             sent_topics_df['Dominant_Topic'] = sent_topics_df.Data.apply(lambda x: x[0])
             sent_topics_df['Perc_Contribution'] = sent_topics_df.Data.apply(lambda x: round(x[1]
             sent_topics_df['Topic_Keywords'] = sent_topics_df.Dominant_Topic.apply(lambda x: top

             sent_topics_df = sent_topics_df[['Dominant_Topic', 'Perc_Contribution', 'Topic_Keywo
             return(sent_topics_df)

         vendor = df2[RAT]['vendor']
         sales = df2[RAT]['total_sales']
         df2_topic_sents_keywords_RAT = format_topics_sentences(ldamodel=optimal_model_RAT, corpu

         # Format
         df2_dominant_topic_RAT = df2_topic_sents_keywords_RAT.reset_index()
         df2_dominant_topic_RAT.columns = ['Document_No', 'Dominant_Topic', 'Topic_Perc_Contrib',

In [34]: # Group top 5 sentences under each topic
         sent_topics_sorted_RAT = pd.DataFrame()

         sent_topics_outdf_grpd = df2_topic_sents_keywords_RAT.groupby('Dominant_Topic')

         for i, grp in sent_topics_outdf_grpd:
```

```
        sent_topics_sorted_RAT = pd.concat([sent_topics_sorted_RAT,
                                            grp.sort_values(['Perc_Contribution'], ascending=
                                           axis=0)

    # Reset Index
    sent_topics_sorted_RAT.reset_index(drop=True, inplace=True)

    # Format
    sent_topics_sorted_RAT.columns = ['Topic_Num', "Topic_Perc_Contrib", "Keywords", "Text", "Inde
```

In [35]:
```
cluster0 = df2_dominant_topic_RAT.loc[df2_dominant_topic_RAT['Dominant_Topic'] == 0]
cluster1 = df2_dominant_topic_RAT.loc[df2_dominant_topic_RAT['Dominant_Topic'] == 1]
cluster2 = df2_dominant_topic_RAT.loc[df2_dominant_topic_RAT['Dominant_Topic'] == 2]
cluster3 = df2_dominant_topic_RAT.loc[df2_dominant_topic_RAT['Dominant_Topic'] == 3]
cluster4 = df2_dominant_topic_RAT.loc[df2_dominant_topic_RAT['Dominant_Topic'] == 4]
cluster5 = df2_dominant_topic_RAT.loc[df2_dominant_topic_RAT['Dominant_Topic'] == 5]
cluster6 = df2_dominant_topic_RAT.loc[df2_dominant_topic_RAT['Dominant_Topic'] == 6]


d = {'Cluster_No' : pd.Series([0,1,2,3,4,5,6]),
     'Total_listings' : pd.Series([len(df2_dominant_topic_RAT)]),
     'Numb_of_list ' : pd.Series([len(cluster0), len(cluster1), len(cluster2), len(cluster3),
     'Total_uniq_vendors' : pd.Series([df2_dominant_topic_RAT.Vendors.nunique()]),
     'Unique_vendors' : pd.Series([cluster0.Vendors.nunique(),cluster1.Vendors.nunique(),clust
     'Total_revenue' : pd.Series([df2_dominant_topic_RAT.Total_sales.sum()]),
     'Revenue' : pd.Series([cluster0.Total_sales.sum(),cluster1.Total_sales.sum(),cluster2.Tot
df_intel = pd.DataFrame(d)
cols = df_intel.columns.tolist()
cols = ['Cluster_No',
 'Total_listings',
 'Numb_of_list ',
 'Total_uniq_vendors',
 'Unique_vendors',
 'Total_revenue',
 'Revenue']
df_intel = df_intel[cols]
```

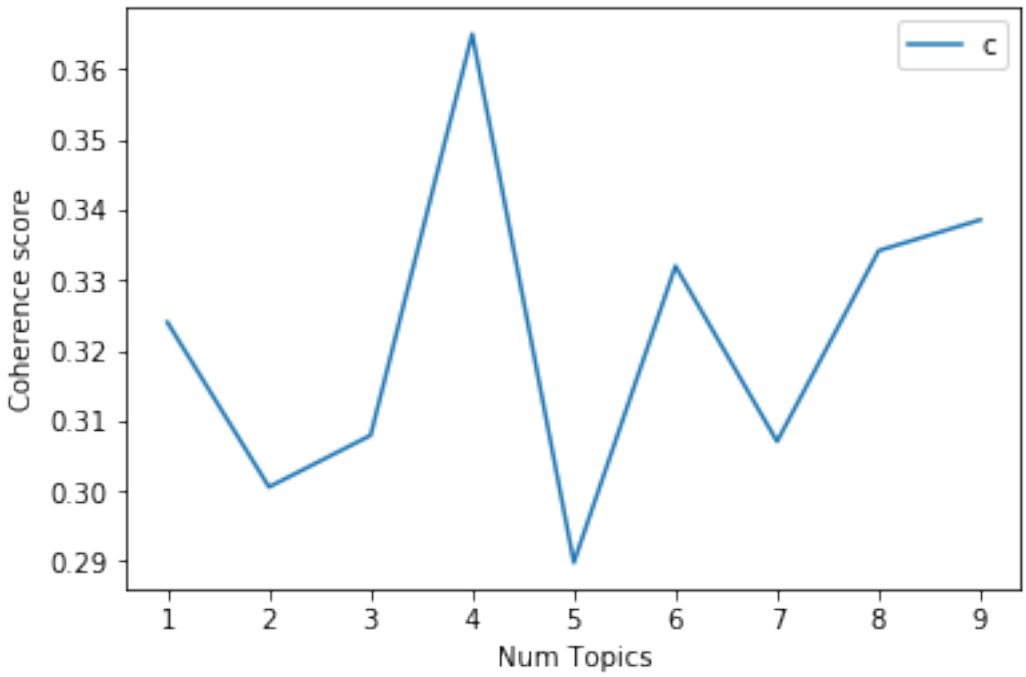
In [38]:
```
writer = ExcelWriter('D:/fegmiedema/Desktop/RAT.xlsx')
sent_topics_sorted_RAT.to_excel(writer,'Sheet1',index=False)
df_intel.to_excel(writer,'Sheet2',index=False)
writer.save()
```

B.2.4. Botnet - 127 listings - 4 topics

In [39]:
```
data_lemmatized_botnet = lemmatization(df2[botnet]['title&description'])
id2word_botnet = corpora.Dictionary(data_lemmatized_botnet)
texts_botnet = data_lemmatized_botnet
corpus_botnet = [id2word_botnet.doc2bow(text) for text in texts_botnet]
```

In [40]:
```
model_list_botnet, coherence_values_botnet = compute_coherence_values(dictionary=id2word_botne
```

In [41]:
```
# Show graph
limit=10; start=1; step=1;
x = range(start, limit, step)
plt.plot(x, coherence_values_botnet)
plt.xlabel("Num Topics")
plt.ylabel("Coherence score")
plt.legend(("coherence_values"), loc='best')
plt.show()
```




```
In [42]: # Print the coherence scores
         for m, cv in zip(x, coherence_values_botnet):
             print("Num Topics =", m, " has Coherence Value of", round(cv, 4))

Num Topics = 1  has Coherence Value of 0.324
Num Topics = 2  has Coherence Value of 0.3006
Num Topics = 3  has Coherence Value of 0.3079
Num Topics = 4  has Coherence Value of 0.365
Num Topics = 5  has Coherence Value of 0.2898
Num Topics = 6  has Coherence Value of 0.332
Num Topics = 7  has Coherence Value of 0.307
Num Topics = 8  has Coherence Value of 0.3342
Num Topics = 9  has Coherence Value of 0.3386
```

```
In [44]: optimal_model_botnet = model_list_botnet[3]
         vis = pyLDAvis.gensim.prepare(optimal_model_botnet, corpus_botnet, id2word_botnet)
         pyLDAvis.display(vis)

Out[44]: <IPython.core.display.HTML object>
```

```
In [45]: def format_topics_sentences(ldamodel, corpus, texts, original, vendor, sales):
             # Array of top 10 topics
             top10array = []

             for row in range(ldamodel.num_topics):
                 wp = ldamodel.show_topic(row)
                 topic_keywords = ", ".join([word for word, prop in wp])
                 top10array.append((row, topic_keywords))

             top10dict = dict(top10array)
```

```
        sent_topics_df = pd.DataFrame(pd.DataFrame([sorted(topic[0], key=lambda x: (x[1]), reverse
        sent_topics_df.columns=["Data"]

        original = original.reset_index()
        sent_topics_df = pd.concat([sent_topics_df, original], axis=1)
        sent_topics_df.columns.values[2] = 'Original'

        vendors = vendor.reset_index()
        sent_topics_df = pd.concat([sent_topics_df, vendors], axis=1)
        sent_topics_df.columns.values[4] = 'Vendors'

        total_sales = sales.reset_index()
        sent_topics_df = pd.concat([sent_topics_df, total_sales], axis=1)
        sent_topics_df.columns.values[5] = 'Total_sales'

        sent_topics_df['Original'] = sent_topics_df.Original
        sent_topics_df['Vendors'] = sent_topics_df.Vendors
        sent_topics_df['Total_sales'] = sent_topics_df.Total_sales
        sent_topics_df['Index'] = sent_topics_df.index
        sent_topics_df['Dominant_Topic'] = sent_topics_df.Data.apply(lambda x: x[0])
        sent_topics_df['Perc_Contribution'] = sent_topics_df.Data.apply(lambda x: round(x[1],4))
        sent_topics_df['Topic_Keywords'] = sent_topics_df.Dominant_Topic.apply(lambda x: top10dict

        sent_topics_df = sent_topics_df[['Dominant_Topic', 'Perc_Contribution', 'Topic_Keywords',
        return(sent_topics_df)

    vendor = df2[botnet]['vendor']
    sales = df2[botnet]['total_sales']
    df2_topic_sents_keywords_botnet = format_topics_sentences(ldamodel=optimal_model_botnet, corpu

    # Format
    df2_dominant_topic_botnet = df2_topic_sents_keywords_botnet.reset_index()
    df2_dominant_topic_botnet.columns = ['Document_No', 'Dominant_Topic', 'Topic_Perc_Contrib', 'K
```

In [46]:
```
    # Group top 5 sentences under each topic
    sent_topics_sorted_botnet = pd.DataFrame()

    sent_topics_outdf_grpd = df2_topic_sents_keywords_botnet.groupby('Dominant_Topic')

    for i, grp in sent_topics_outdf_grpd:
        sent_topics_sorted_botnet = pd.concat([sent_topics_sorted_botnet,
                                               grp.sort_values(['Perc_Contribution'], ascending=
                                              axis=0)


    # Reset Index
    sent_topics_sorted_botnet.reset_index(drop=True, inplace=True)

    # Format
    sent_topics_sorted_botnet.columns = ['Topic_Num', "Topic_Perc_Contrib", "Keywords", "Text", "I
```

In [47]:
```
    cluster0 = df2_dominant_topic_botnet.loc[df2_dominant_topic_botnet['Dominant_Topic'] == 0]
    cluster1 = df2_dominant_topic_botnet.loc[df2_dominant_topic_botnet['Dominant_Topic'] == 1]
    cluster2 = df2_dominant_topic_botnet.loc[df2_dominant_topic_botnet['Dominant_Topic'] == 2]
    cluster3 = df2_dominant_topic_botnet.loc[df2_dominant_topic_botnet['Dominant_Topic'] == 3]

    d = {'Cluster_No' : pd.Series([0,1,2,3,4,5]),
```

```
        'Total_listings' : pd.Series([len(df2_dominant_topic_botnet)]),
        'Numb_of_list ' : pd.Series([len(cluster0), len(cluster1), len(cluster2), len(clust
        'Total_uniq_vendors' : pd.Series([df2_dominant_topic_botnet.Vendors.nunique()]),
        'Unique_vendors' : pd.Series([cluster0.Vendors.nunique(),cluster1.Vendors.nunique()
        'Total_revenue' : pd.Series([df2_dominant_topic_botnet.Total_sales.sum()]),
        'Revenue' : pd.Series([cluster0.Total_sales.sum(),cluster1.Total_sales.sum(),cluste
df_intel = pd.DataFrame(d)
cols = df_intel.columns.tolist()
cols = ['Cluster_No',
 'Total_listings',
 'Numb_of_list ',
 'Total_uniq_vendors',
 'Unique_vendors',
 'Total_revenue',
 'Revenue']
df_intel = df_intel[cols]
```

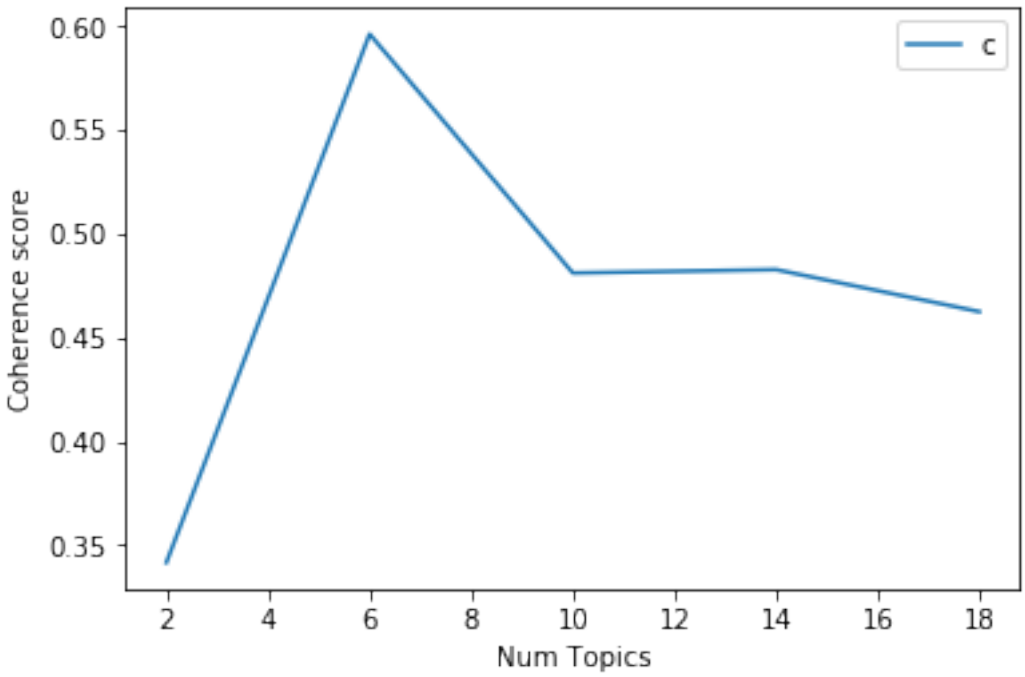
```
In [48]: writer = ExcelWriter('D:/fegmiedema/Desktop/Botnet.xlsx')
         sent_topics_sorted_botnet.to_excel(writer,'Sheet1',index=False)
         df_intel.to_excel(writer,'Sheet2',index=False)
         writer.save()
```

B.2.5. Email - 562 listings - 6 topics

```
In [49]: data_lemmatized_email = lemmatization(df2[email]['title&description'])
         id2word_email = corpora.Dictionary(data_lemmatized_email)
         texts_email = data_lemmatized_email
         corpus_email = [id2word_email.doc2bow(text) for text in texts_email]
```

```
In [50]: model_list_email, coherence_values_email = compute_coherence_values(dictionary=id2word_e
```

```
In [51]: # Show graph
         limit=20; start=2; step=4;
         x = range(start, limit, step)
         plt.plot(x, coherence_values_email)
         plt.xlabel("Num Topics")
         plt.ylabel("Coherence score")
         plt.legend(("coherence_values"), loc='best')
         plt.show()
```




```
In [52]: # Print the coherence scores
         for m, cv in zip(x, coherence_values_email):
             print("Num Topics =", m, " has Coherence Value of", round(cv, 4))
```

```
Num Topics = 2  has Coherence Value of 0.3417
Num Topics = 6  has Coherence Value of 0.5961
Num Topics = 10  has Coherence Value of 0.4811
Num Topics = 14  has Coherence Value of 0.4827
Num Topics = 18  has Coherence Value of 0.4625
```

```
In [53]: optimal_model_email = model_list_email[1]
         vis = pyLDAvis.gensim.prepare(optimal_model_email, corpus_email, id2word_email)
         pyLDAvis.display(vis)
```

```
Out[53]: <IPython.core.display.HTML object>
```

```
In [54]: def format_topics_sentences(ldamodel, corpus, texts, original, vendor, sales):
             # Array of top 10 topics
             top10array = []

             for row in range(ldamodel.num_topics):
                 wp = ldamodel.show_topic(row)
                 topic_keywords = ", ".join([word for word, prop in wp])
                 top10array.append((row, topic_keywords))

             top10dict = dict(top10array)

             sent_topics_df = pd.DataFrame(pd.DataFrame([sorted(topic[0], key=lambda x: (x[1]), reverse
             sent_topics_df.columns=["Data"]
```

```
        original = original.reset_index()
        sent_topics_df = pd.concat([sent_topics_df, original], axis=1)
        sent_topics_df.columns.values[2] = 'Original'

        vendors = vendor.reset_index()
        sent_topics_df = pd.concat([sent_topics_df, vendors], axis=1)
        sent_topics_df.columns.values[4] = 'Vendors'

        total_sales = sales.reset_index()
        sent_topics_df = pd.concat([sent_topics_df, total_sales], axis=1)
        sent_topics_df.columns.values[5] = 'Total_sales'

        sent_topics_df['Original'] = sent_topics_df.Original
        sent_topics_df['Vendors'] = sent_topics_df.Vendors
        sent_topics_df['Total_sales'] = sent_topics_df.Total_sales
        sent_topics_df['Index'] = sent_topics_df.index
        sent_topics_df['Dominant_Topic'] = sent_topics_df.Data.apply(lambda x: x[0])
        sent_topics_df['Perc_Contribution'] = sent_topics_df.Data.apply(lambda x: round(x[1]
        sent_topics_df['Topic_Keywords'] = sent_topics_df.Dominant_Topic.apply(lambda x: top

        sent_topics_df = sent_topics_df[['Dominant_Topic', 'Perc_Contribution', 'Topic_Keywo
        return(sent_topics_df)

    vendor = df2[email]['vendor']
    sales = df2[email]['total_sales']
    df2_topic_sents_keywords_email = format_topics_sentences(ldamodel=optimal_model_email, c

    # Format
    df2_dominant_topic_email = df2_topic_sents_keywords_email.reset_index()
    df2_dominant_topic_email.columns = ['Document_No', 'Dominant_Topic', 'Topic_Perc_Contrib
```

In [55]:
```
    # Group top 5 sentences under each topic
    sent_topics_sorted_email = pd.DataFrame()

    sent_topics_outdf_grpd = df2_topic_sents_keywords_email.groupby('Dominant_Topic')

    for i, grp in sent_topics_outdf_grpd:
        sent_topics_sorted_email = pd.concat([sent_topics_sorted_email,
                                              grp.sort_values(['Perc_Contribution'], asce
                                             axis=0)

    # Reset Index
    sent_topics_sorted_email.reset_index(drop=True, inplace=True)

    # Format
    sent_topics_sorted_email.columns = ['Topic_Num', "Topic_Perc_Contrib", "Keywords", "Text
```

In [56]:
```
    cluster0 = df2_dominant_topic_email.loc[df2_dominant_topic_email['Dominant_Topic'] == 0]
    cluster1 = df2_dominant_topic_email.loc[df2_dominant_topic_email['Dominant_Topic'] == 1]
    cluster2 = df2_dominant_topic_email.loc[df2_dominant_topic_email['Dominant_Topic'] == 2]
    cluster3 = df2_dominant_topic_email.loc[df2_dominant_topic_email['Dominant_Topic'] == 3]
    cluster4 = df2_dominant_topic_email.loc[df2_dominant_topic_email['Dominant_Topic'] == 4]
    cluster5 = df2_dominant_topic_email.loc[df2_dominant_topic_email['Dominant_Topic'] == 5]


    d = {'Cluster_No' : pd.Series([0,1,2,3,4,5]),
         'Total_listings' : pd.Series([len(df2_dominant_topic_email)]),
```

```
        'Numb_of_list ' : pd.Series([len(cluster0), len(cluster1), len(cluster2), len(cluster3),
        'Total_uniq_vendors' : pd.Series([df2_dominant_topic_email.Vendors.nunique()]),
        'Unique_vendors' : pd.Series([cluster0.Vendors.nunique(),cluster1.Vendors.nunique(),clust
        'Total_revenue' : pd.Series([df2_dominant_topic_email.Total_sales.sum()]),
        'Revenue' : pd.Series([cluster0.Total_sales.sum(),cluster1.Total_sales.sum(),cluster2.Tot
df_intel = pd.DataFrame(d)
cols = df_intel.columns.tolist()
cols = ['Cluster_No',
 'Total_listings',
 'Numb_of_list ',
 'Total_uniq_vendors',
 'Unique_vendors',
 'Total_revenue',
 'Revenue']
df_intel = df_intel[cols]
```

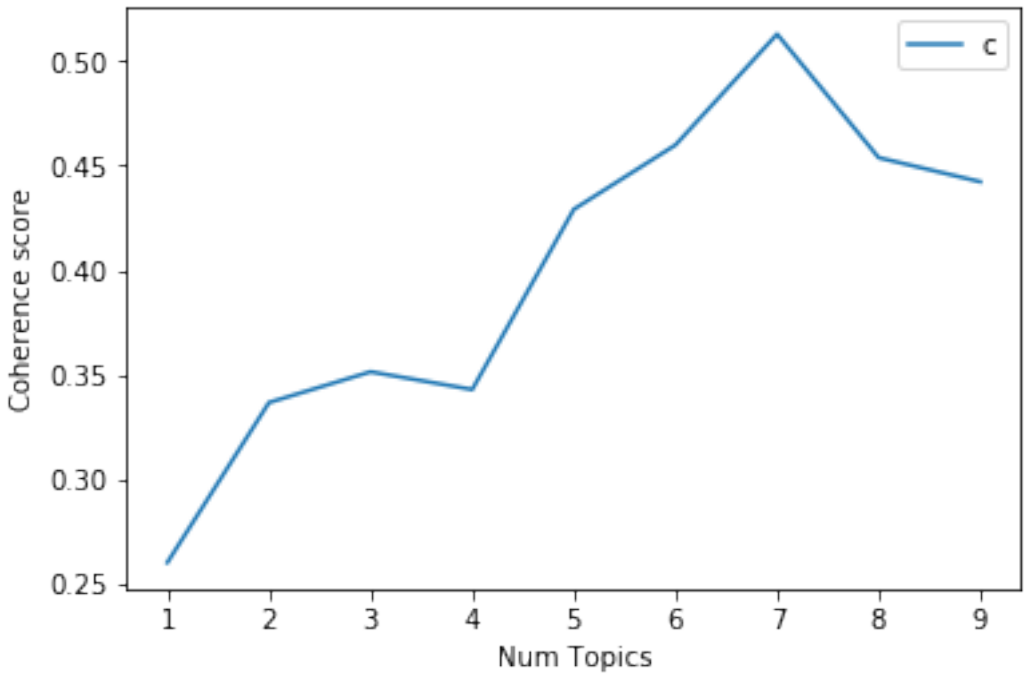
```
In [57]: writer = ExcelWriter('D:/fegmiedema/Desktop/Email.xlsx')
         sent_topics_sorted_email.to_excel(writer,'Sheet1',index=False)
         df_intel.to_excel(writer,'Sheet2',index=False)
         writer.save()
```

B.2.6. Exploits - 117 listings - 7 topics

```
In [58]: data_lemmatized_exploits = lemmatization(df2[exploits]['title&description'])
         id2word_exploits = corpora.Dictionary(data_lemmatized_exploits)
         texts_exploits = data_lemmatized_exploits
         corpus_exploits = [id2word_exploits.doc2bow(text) for text in texts_exploits]
```

```
In [61]: model_list_exploits, coherence_values_exploits = compute_coherence_values(dictionary=id2word_e
```

```
In [62]: # Show graph
         limit=10; start=1; step=1;
         x = range(start, limit, step)
         plt.plot(x, coherence_values_exploits)
         plt.xlabel("Num Topics")
         plt.ylabel("Coherence score")
         plt.legend(("coherence_values"), loc='best')
         plt.show()
```




```
In [63]: # Print the coherence scores
         for m, cv in zip(x, coherence_values_exploits):
             print("Num Topics =", m, " has Coherence Value of", round(cv, 4))
```

```
Num Topics = 1  has Coherence Value of 0.2604
Num Topics = 2  has Coherence Value of 0.3368
Num Topics = 3  has Coherence Value of 0.3515
Num Topics = 4  has Coherence Value of 0.343
Num Topics = 5  has Coherence Value of 0.429
Num Topics = 6  has Coherence Value of 0.4597
Num Topics = 7  has Coherence Value of 0.5127
Num Topics = 8  has Coherence Value of 0.4537
Num Topics = 9  has Coherence Value of 0.4422
```

```
In [64]: optimal_model_exploits = model_list_exploits[6]
         vis = pyLDAvis.gensim.prepare(optimal_model_exploits, corpus_exploits, id2word_exploits)
         pyLDAvis.display(vis)
```

```
Out[64]: <IPython.core.display.HTML object>
```

```
In [65]: def format_topics_sentences(ldamodel, corpus, texts, original, vendor, sales):
             # Array of top 10 topics
             top10array = []

             for row in range(ldamodel.num_topics):
                 wp = ldamodel.show_topic(row)
                 topic_keywords = ", ".join([word for word, prop in wp])
                 top10array.append((row, topic_keywords))

             top10dict = dict(top10array)
```

```
        sent_topics_df = pd.DataFrame(pd.DataFrame([sorted(topic[0], key=lambda x: (x[1]), reverse
        sent_topics_df.columns=["Data"]

        original = original.reset_index()
        sent_topics_df = pd.concat([sent_topics_df, original], axis=1)
        sent_topics_df.columns.values[2] = 'Original'

        vendors = vendor.reset_index()
        sent_topics_df = pd.concat([sent_topics_df, vendors], axis=1)
        sent_topics_df.columns.values[4] = 'Vendors'

        total_sales = sales.reset_index()
        sent_topics_df = pd.concat([sent_topics_df, total_sales], axis=1)
        sent_topics_df.columns.values[5] = 'Total_sales'

        sent_topics_df['Original'] = sent_topics_df.Original
        sent_topics_df['Vendors'] = sent_topics_df.Vendors
        sent_topics_df['Total_sales'] = sent_topics_df.Total_sales
        sent_topics_df['Index'] = sent_topics_df.index
        sent_topics_df['Dominant_Topic'] = sent_topics_df.Data.apply(lambda x: x[0])
        sent_topics_df['Perc_Contribution'] = sent_topics_df.Data.apply(lambda x: round(x[1],4))
        sent_topics_df['Topic_Keywords'] = sent_topics_df.Dominant_Topic.apply(lambda x: top10dict

        sent_topics_df = sent_topics_df[['Dominant_Topic', 'Perc_Contribution', 'Topic_Keywords',
        return(sent_topics_df)

vendor = df2[exploits]['vendor']
sales = df2[exploits]['total_sales']
df2_topic_sents_keywords_exploits = format_topics_sentences(ldamodel=optimal_model_exploits, c

# Format
df2_dominant_topic_exploits = df2_topic_sents_keywords_exploits.reset_index()
df2_dominant_topic_exploits.columns = ['Document_No', 'Dominant_Topic', 'Topic_Perc_Contrib',
```

In [66]:
```
# Group top 5 sentences under each topic
sent_topics_sorted_exploits = pd.DataFrame()

sent_topics_outdf_grpd = df2_topic_sents_keywords_exploits.groupby('Dominant_Topic')

for i, grp in sent_topics_outdf_grpd:
    sent_topics_sorted_exploits = pd.concat([sent_topics_sorted_exploits,
                                             grp.sort_values(['Perc_Contribution'], ascending=
                                            axis=0)


# Reset Index
sent_topics_sorted_exploits.reset_index(drop=True, inplace=True)

# Format
sent_topics_sorted_exploits.columns = ['Topic_Num', "Topic_Perc_Contrib", "Keywords", "Text",
```

In [67]:
```
cluster0 = df2_dominant_topic_exploits.loc[df2_dominant_topic_exploits['Dominant_Topic'] == 0]
cluster1 = df2_dominant_topic_exploits.loc[df2_dominant_topic_exploits['Dominant_Topic'] == 1]
cluster2 = df2_dominant_topic_exploits.loc[df2_dominant_topic_exploits['Dominant_Topic'] == 2]
cluster3 = df2_dominant_topic_exploits.loc[df2_dominant_topic_exploits['Dominant_Topic'] == 3]
cluster4 = df2_dominant_topic_exploits.loc[df2_dominant_topic_exploits['Dominant_Topic'] == 4]
cluster5 = df2_dominant_topic_exploits.loc[df2_dominant_topic_exploits['Dominant_Topic'] == 5]
```

```
cluster6 = df2_dominant_topic_exploits.loc[df2_dominant_topic_exploits['Dominant_Topic']

d = {'Cluster_No' : pd.Series([0,1,2,3,4,5,6]),
     'Total_listings' : pd.Series([len(df2_dominant_topic_exploits)]),
     'Numb_of_list ' : pd.Series([len(cluster0), len(cluster1), len(cluster2), len(clust
     'Total_uniq_vendors' : pd.Series([df2_dominant_topic_exploits.Vendors.nunique()]),
     'Unique_vendors' : pd.Series([cluster0.Vendors.nunique(),cluster1.Vendors.nunique()
     'Total_revenue' : pd.Series([df2_dominant_topic_exploits.Total_sales.sum()]),
     'Revenue' : pd.Series([cluster0.Total_sales.sum(),cluster1.Total_sales.sum(),cluste
df_intel = pd.DataFrame(d)
cols = df_intel.columns.tolist()
cols = ['Cluster_No',
 'Total_listings',
 'Numb_of_list ',
 'Total_uniq_vendors',
 'Unique_vendors',
 'Total_revenue',
 'Revenue']
df_intel = df_intel[cols]
```

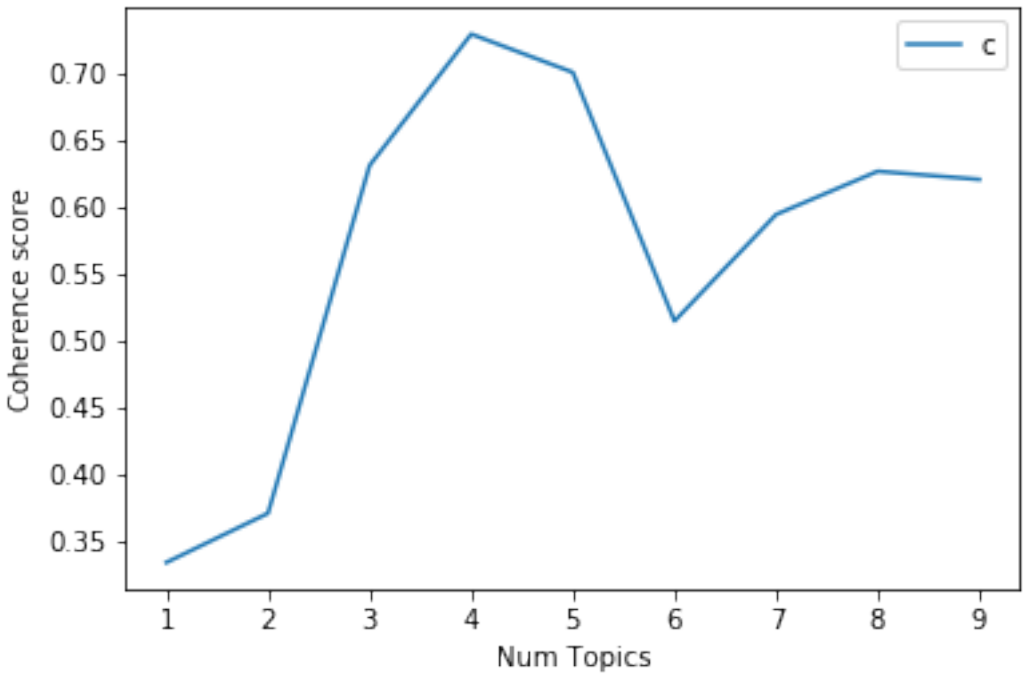
```
In [68]: writer = ExcelWriter('D:/fegmiedema/Desktop/Exploits.xlsx')
         sent_topics_sorted_exploits.to_excel(writer,'Sheet1',index=False)
         df_intel.to_excel(writer,'Sheet2',index=False)
         writer.save()
```

B.2.7. Hosting - 20 listings - 4 topics

```
In [69]: data_lemmatized_hosting = lemmatization(df2[hosting]['title&description'])
         id2word_hosting = corpora.Dictionary(data_lemmatized_hosting)
         texts_hosting = data_lemmatized_hosting
         corpus_hosting = [id2word_hosting.doc2bow(text) for text in texts_hosting]
```

```
In [70]: model_list_hosting, coherence_values_hosting = compute_coherence_values(dictionary=id2wo
```

```
In [71]: # Show graph
         limit=10; start=1; step=1;
         x = range(start, limit, step)
         plt.plot(x, coherence_values_hosting)
         plt.xlabel("Num Topics")
         plt.ylabel("Coherence score")
         plt.legend(("coherence_values"), loc='best')
         plt.show()
```




```
In [72]: # Print the coherence scores
         for m, cv in zip(x, coherence_values_hosting):
             print("Num Topics =", m, " has Coherence Value of", round(cv, 4))
```

```
Num Topics = 1  has Coherence Value of 0.3345
Num Topics = 2  has Coherence Value of 0.3712
Num Topics = 3  has Coherence Value of 0.6307
Num Topics = 4  has Coherence Value of 0.7287
Num Topics = 5  has Coherence Value of 0.7002
Num Topics = 6  has Coherence Value of 0.5146
Num Topics = 7  has Coherence Value of 0.5941
Num Topics = 8  has Coherence Value of 0.6264
Num Topics = 9  has Coherence Value of 0.6203
```

```
In [73]: optimal_model_hosting = model_list_hosting[3]
         vis = pyLDAvis.gensim.prepare(optimal_model_hosting, corpus_hosting, id2word_hosting)
         pyLDAvis.display(vis)
```

```
Out[73]: <IPython.core.display.HTML object>
```

```
In [74]: def format_topics_sentences(ldamodel, corpus, texts, original, vendor, sales):
             # Array of top 10 topics
             top10array = []

             for row in range(ldamodel.num_topics):
                 wp = ldamodel.show_topic(row)
                 topic_keywords = ", ".join([word for word, prop in wp])
                 top10array.append((row, topic_keywords))

             top10dict = dict(top10array)
```

```
        sent_topics_df = pd.DataFrame(pd.DataFrame([sorted(topic[0], key=lambda x: (x[1]), r
        sent_topics_df.columns=["Data"]

        original = original.reset_index()
        sent_topics_df = pd.concat([sent_topics_df, original], axis=1)
        sent_topics_df.columns.values[2] = 'Original'

        vendors = vendor.reset_index()
        sent_topics_df = pd.concat([sent_topics_df, vendors], axis=1)
        sent_topics_df.columns.values[4] = 'Vendors'

        total_sales = sales.reset_index()
        sent_topics_df = pd.concat([sent_topics_df, total_sales], axis=1)
        sent_topics_df.columns.values[5] = 'Total_sales'

        sent_topics_df['Original'] = sent_topics_df.Original
        sent_topics_df['Vendors'] = sent_topics_df.Vendors
        sent_topics_df['Total_sales'] = sent_topics_df.Total_sales
        sent_topics_df['Index'] = sent_topics_df.index
        sent_topics_df['Dominant_Topic'] = sent_topics_df.Data.apply(lambda x: x[0])
        sent_topics_df['Perc_Contribution'] = sent_topics_df.Data.apply(lambda x: round(x[1]
        sent_topics_df['Topic_Keywords'] = sent_topics_df.Dominant_Topic.apply(lambda x: top

        sent_topics_df = sent_topics_df[['Dominant_Topic', 'Perc_Contribution', 'Topic_Keywo
        return(sent_topics_df)

    vendor = df2[hosting]['vendor']
    sales = df2[hosting]['total_sales']
    df2_topic_sents_keywords_hosting = format_topics_sentences(ldamodel=optimal_model_hostin

    # Format
    df2_dominant_topic_hosting = df2_topic_sents_keywords_hosting.reset_index()
    df2_dominant_topic_hosting.columns = ['Document_No', 'Dominant_Topic', 'Topic_Perc_Contr
```

In [75]:
```
    # Group top 5 sentences under each topic
    sent_topics_sorted_hosting = pd.DataFrame()

    sent_topics_outdf_grpd = df2_topic_sents_keywords_hosting.groupby('Dominant_Topic')

    for i, grp in sent_topics_outdf_grpd:
        sent_topics_sorted_hosting = pd.concat([sent_topics_sorted_hosting,
                                                grp.sort_values(['Perc_Contribution'], asce
                                               axis=0)


    # Reset Index
    sent_topics_sorted_hosting.reset_index(drop=True, inplace=True)

    # Format
    sent_topics_sorted_hosting.columns = ['Topic_Num', "Topic_Perc_Contrib", "Keywords", "Te
```

In [76]:
```
    cluster0 = df2_dominant_topic_hosting.loc[df2_dominant_topic_hosting['Dominant_Topic'] =
    cluster1 = df2_dominant_topic_hosting.loc[df2_dominant_topic_hosting['Dominant_Topic'] =
    cluster2 = df2_dominant_topic_hosting.loc[df2_dominant_topic_hosting['Dominant_Topic'] =
    cluster3 = df2_dominant_topic_hosting.loc[df2_dominant_topic_hosting['Dominant_Topic'] =

    d = {'Cluster_No' : pd.Series([0,1,2,3]),
```

```
        'Total_listings' : pd.Series([len(df2_dominant_topic_hosting)]),
        'Numb_of_list ' : pd.Series([len(cluster0), len(cluster1), len(cluster2), len(cluster3)]),
        'Total_uniq_vendors' : pd.Series([df2_dominant_topic_hosting.Vendors.nunique()]),
        'Unique_vendors' : pd.Series([cluster0.Vendors.nunique(),cluster1.Vendors.nunique(),clust
        'Total_revenue' : pd.Series([df2_dominant_topic_hosting.Total_sales.sum()]),
        'Revenue' : pd.Series([cluster0.Total_sales.sum(),cluster1.Total_sales.sum(),cluster2.Tot
    df_intel = pd.DataFrame(d)
    cols = df_intel.columns.tolist()
    cols = ['Cluster_No',
     'Total_listings',
     'Numb_of_list ',
     'Total_uniq_vendors',
     'Unique_vendors',
     'Total_revenue',
     'Revenue']
    df_intel = df_intel[cols]
```

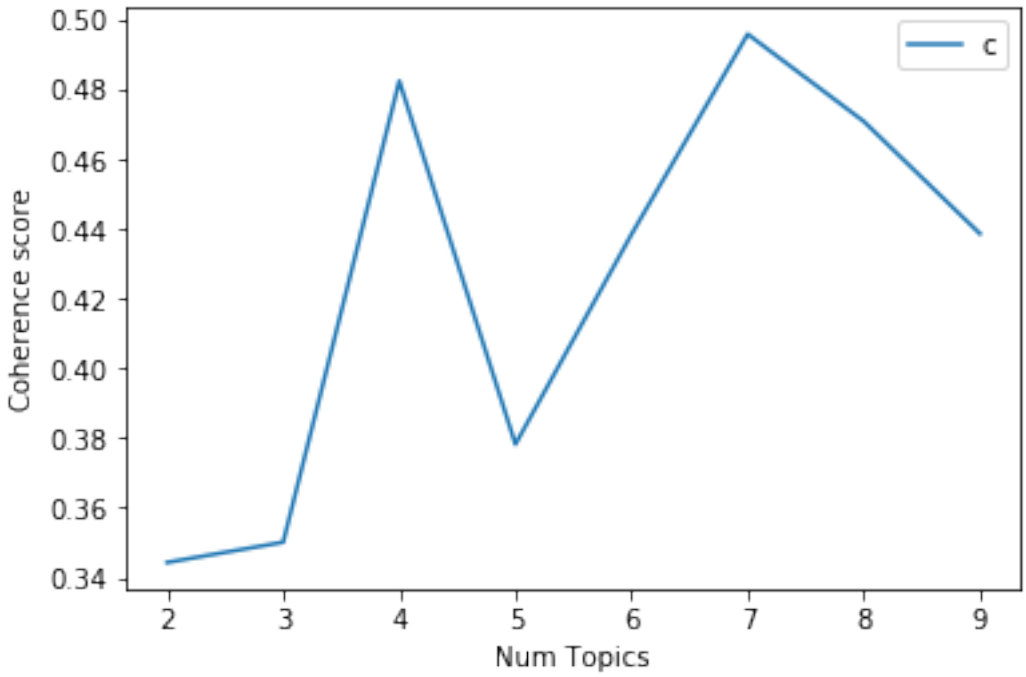
```
In [77]: writer = ExcelWriter('D:/fegmiedema/Desktop/Hosting.xlsx')
         sent_topics_sorted_hosting.to_excel(writer,'Sheet1',index=False)
         df_intel.to_excel(writer,'Sheet2',index=False)
         writer.save()
```

B.2.8. Malware - 316 listings - 7 topics

```
In [78]: data_lemmatized_malware = lemmatization(df2[malware]['title&description'])
         id2word_malware = corpora.Dictionary(data_lemmatized_malware)
         texts_malware = data_lemmatized_malware
         corpus_malware = [id2word_malware.doc2bow(text) for text in texts_malware]
```

```
In [79]: model_list_malware, coherence_values_malware = compute_coherence_values(dictionary=id2word_mal
```

```
In [80]: # Show graph
         limit=10; start=2; step=1;
         x = range(start, limit, step)
         plt.plot(x, coherence_values_malware)
         plt.xlabel("Num Topics")
         plt.ylabel("Coherence score")
         plt.legend(("coherence_values"), loc='best')
         plt.show()
```




```
In [81]: # Print the coherence scores
         for m, cv in zip(x, coherence_values_malware):
             print("Num Topics =", m, " has Coherence Value of", round(cv, 4))
```

```
Num Topics = 2  has Coherence Value of 0.3443
Num Topics = 3  has Coherence Value of 0.3501
Num Topics = 4  has Coherence Value of 0.4826
Num Topics = 5  has Coherence Value of 0.3782
Num Topics = 6  has Coherence Value of 0.4386
Num Topics = 7  has Coherence Value of 0.496
Num Topics = 8  has Coherence Value of 0.471
Num Topics = 9  has Coherence Value of 0.4387
```

```
In [86]: optimal_model_malware = model_list_malware[5]
         vis = pyLDAvis.gensim.prepare(optimal_model_malware, corpus_malware, id2word_malware)
         pyLDAvis.display(vis)
```

```
Out[86]: <IPython.core.display.HTML object>
```

```
In [87]: def format_topics_sentences(ldamodel, corpus, texts, original, vendor, sales):
             # Array of top 10 topics
             top10array = []

             for row in range(ldamodel.num_topics):
                 wp = ldamodel.show_topic(row)
                 topic_keywords = ", ".join([word for word, prop in wp])
                 top10array.append((row, topic_keywords))

             top10dict = dict(top10array)
```

```
        sent_topics_df = pd.DataFrame(pd.DataFrame([sorted(topic[0], key=lambda x: (x[1]), reverse
        sent_topics_df.columns=["Data"]

        original = original.reset_index()
        sent_topics_df = pd.concat([sent_topics_df, original], axis=1)
        sent_topics_df.columns.values[2] = 'Original'

        vendors = vendor.reset_index()
        sent_topics_df = pd.concat([sent_topics_df, vendors], axis=1)
        sent_topics_df.columns.values[4] = 'Vendors'

        total_sales = sales.reset_index()
        sent_topics_df = pd.concat([sent_topics_df, total_sales], axis=1)
        sent_topics_df.columns.values[5] = 'Total_sales'

        sent_topics_df['Original'] = sent_topics_df.Original
        sent_topics_df['Vendors'] = sent_topics_df.Vendors
        sent_topics_df['Total_sales'] = sent_topics_df.Total_sales
        sent_topics_df['Index'] = sent_topics_df.index
        sent_topics_df['Dominant_Topic'] = sent_topics_df.Data.apply(lambda x: x[0])
        sent_topics_df['Perc_Contribution'] = sent_topics_df.Data.apply(lambda x: round(x[1],4))
        sent_topics_df['Topic_Keywords'] = sent_topics_df.Dominant_Topic.apply(lambda x: top10dict

        sent_topics_df = sent_topics_df[['Dominant_Topic', 'Perc_Contribution', 'Topic_Keywords',
        return(sent_topics_df)

    vendor = df2[malware]['vendor']
    sales = df2[malware]['total_sales']
    df2_topic_sents_keywords_malware = format_topics_sentences(ldamodel=optimal_model_malware, cor

    # Format
    df2_dominant_topic_malware = df2_topic_sents_keywords_malware.reset_index()
    df2_dominant_topic_malware.columns = ['Document_No', 'Dominant_Topic', 'Topic_Perc_Contrib', '
```

In [88]:
```
    # Group top 5 sentences under each topic
    sent_topics_sorted_malware = pd.DataFrame()

    sent_topics_outdf_grpd = df2_topic_sents_keywords_malware.groupby('Dominant_Topic')

    for i, grp in sent_topics_outdf_grpd:
        sent_topics_sorted_malware = pd.concat([sent_topics_sorted_malware,
                                                grp.sort_values(['Perc_Contribution'], ascending=
                                               axis=0)


    # Reset Index
    sent_topics_sorted_malware.reset_index(drop=True, inplace=True)

    # Format
    sent_topics_sorted_malware.columns = ['Topic_Num', "Topic_Perc_Contrib", "Keywords", "Text", "
```

In [89]:
```
    cluster0 = df2_dominant_topic_malware.loc[df2_dominant_topic_malware['Dominant_Topic'] == 0]
    cluster1 = df2_dominant_topic_malware.loc[df2_dominant_topic_malware['Dominant_Topic'] == 1]
    cluster2 = df2_dominant_topic_malware.loc[df2_dominant_topic_malware['Dominant_Topic'] == 2]
    cluster3 = df2_dominant_topic_malware.loc[df2_dominant_topic_malware['Dominant_Topic'] == 3]
    cluster4 = df2_dominant_topic_malware.loc[df2_dominant_topic_malware['Dominant_Topic'] == 4]
    cluster5 = df2_dominant_topic_malware.loc[df2_dominant_topic_malware['Dominant_Topic'] == 5]
    cluster6 = df2_dominant_topic_malware.loc[df2_dominant_topic_malware['Dominant_Topic'] == 6]
```

```
d = {'Cluster_No' : pd.Series([0,1,2,3,4,5,6]),
     'Total_listings' : pd.Series([len(df2_dominant_topic_malware)]),
     'Numb_of_list ' : pd.Series([len(cluster0), len(cluster1), len(cluster2), len(clust
     'Total_uniq_vendors' : pd.Series([df2_dominant_topic_malware.Vendors.nunique()]),
     'Unique_vendors' : pd.Series([cluster0.Vendors.nunique(),cluster1.Vendors.nunique()
     'Total_revenue' : pd.Series([df2_dominant_topic_malware.Total_sales.sum()]),
     'Revenue' : pd.Series([cluster0.Total_sales.sum(),cluster1.Total_sales.sum(),cluste
df_intel = pd.DataFrame(d)
cols = df_intel.columns.tolist()
cols = ['Cluster_No',
 'Total_listings',
 'Numb_of_list ',
 'Total_uniq_vendors',
 'Unique_vendors',
 'Total_revenue',
 'Revenue']
df_intel = df_intel[cols]
```

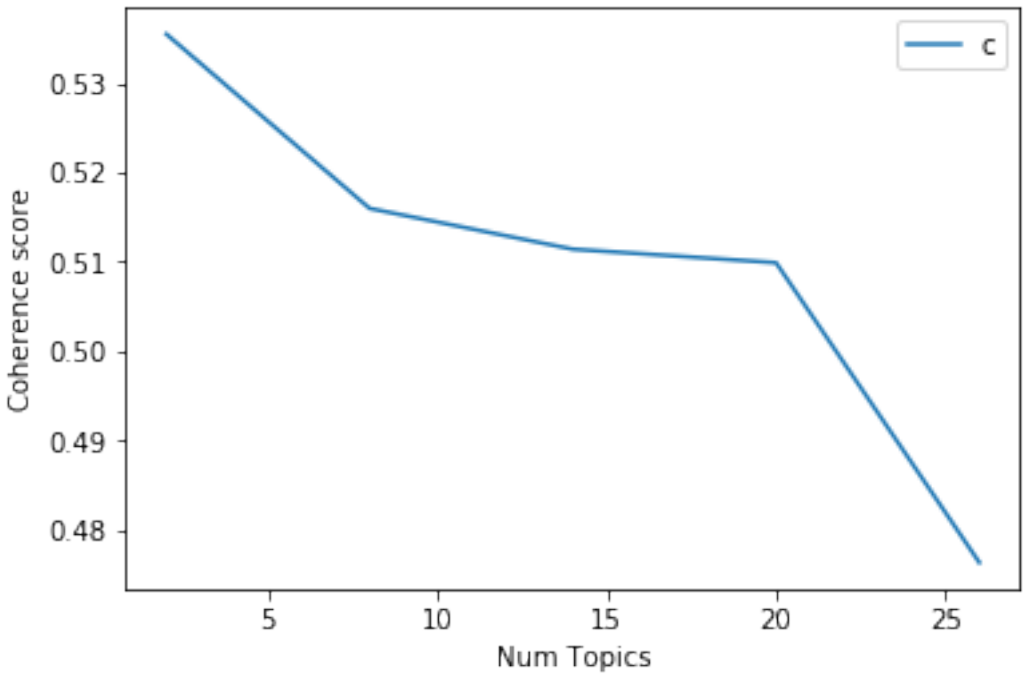
```
In [90]: writer = ExcelWriter('D:/fegmiedema/Desktop/Malware.xlsx')
         sent_topics_sorted_malware.to_excel(writer,'Sheet1',index=False)
         df_intel.to_excel(writer,'Sheet2',index=False)
         writer.save()
```

B.2.9. Other - 8489 listings - 4 topics

```
In [91]: data_lemmatized_other = lemmatization(df2[other]['title&description'])
         id2word_other = corpora.Dictionary(data_lemmatized_other)
         texts_other = data_lemmatized_other
         corpus_other = [id2word_other.doc2bow(text) for text in texts_other]
```

```
In [92]: model_list_other, coherence_values_other = compute_coherence_values(dictionary=id2word_o
```

```
In [93]: # Show graph
         limit=30; start=2; step=6;
         x = range(start, limit, step)
         plt.plot(x, coherence_values_other)
         plt.xlabel("Num Topics")
         plt.ylabel("Coherence score")
         plt.legend(("coherence_values"), loc='best')
         plt.show()
```




```
In [179]: # Print the coherence scores
          for m, cv in zip(x, coherence_values_other):
              print("Num Topics =", m, " has Coherence Value of", round(cv, 4))
```

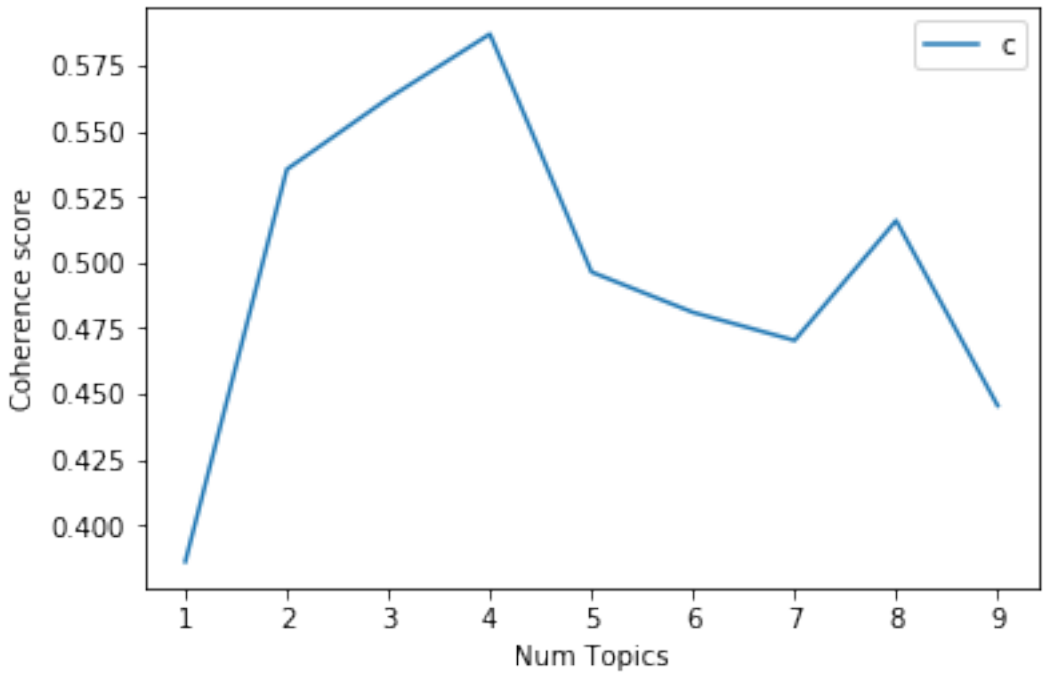
```
Num Topics = 2  has Coherence Value of 0.4352
Num Topics = 8  has Coherence Value of 0.4434
Num Topics = 14  has Coherence Value of 0.4664
Num Topics = 20  has Coherence Value of 0.4685
Num Topics = 26  has Coherence Value of 0.4789
```

```
In [229]: optimal_model_other = model_list_other[1]
          vis = pyLDAvis.gensim.prepare(optimal_model_other, corpus_other, id2word_other)
          pyLDAvis.display(vis)
```

```
Out[229]: <IPython.core.display.HTML object>
```

```
In [96]: model_list_other2, coherence_values_other2 = compute_coherence_values(dictionary=id2word_other
```

```
In [97]: # Show graph
         limit=10; start=1; step=1;
         x = range(start, limit, step)
         plt.plot(x, coherence_values_other2)
         plt.xlabel("Num Topics")
         plt.ylabel("Coherence score")
         plt.legend(("coherence_values"), loc='best')
         plt.show()
```




```
In [98]: # Print the coherence scores
         for m, cv in zip(x, coherence_values_other2):
             print("Num Topics =", m, " has Coherence Value of", round(cv, 4))

Num Topics = 1  has Coherence Value of 0.3861
Num Topics = 2  has Coherence Value of 0.5355
Num Topics = 3  has Coherence Value of 0.5626
Num Topics = 4  has Coherence Value of 0.587
Num Topics = 5  has Coherence Value of 0.4964
Num Topics = 6  has Coherence Value of 0.481
Num Topics = 7  has Coherence Value of 0.4703
Num Topics = 8  has Coherence Value of 0.516
Num Topics = 9  has Coherence Value of 0.4455
```

```
In [99]: optimal_model_other = model_list_other2[3]
         vis = pyLDAvis.gensim.prepare(optimal_model_other, corpus_other, id2word_other)
         pyLDAvis.display(vis)

Out[99]: <IPython.core.display.HTML object>
```

```
In [100]: def format_topics_sentences(ldamodel, corpus, texts, original, vendor, sales):
              # Array of top 10 topics
              top10array = []

              for row in range(ldamodel.num_topics):
                  wp = ldamodel.show_topic(row)
                  topic_keywords = ", ".join([word for word, prop in wp])
                  top10array.append((row, topic_keywords))

              top10dict = dict(top10array)
```

```
        sent_topics_df = pd.DataFrame(pd.DataFrame([sorted(topic[0], key=lambda x: (x[1]), revers
        sent_topics_df.columns=["Data"]

        original = original.reset_index()
        sent_topics_df = pd.concat([sent_topics_df, original], axis=1)
        sent_topics_df.columns.values[2] = 'Original'

        vendors = vendor.reset_index()
        sent_topics_df = pd.concat([sent_topics_df, vendors], axis=1)
        sent_topics_df.columns.values[4] = 'Vendors'

        total_sales = sales.reset_index()
        sent_topics_df = pd.concat([sent_topics_df, total_sales], axis=1)
        sent_topics_df.columns.values[5] = 'Total_sales'

        sent_topics_df['Original'] = sent_topics_df.Original
        sent_topics_df['Vendors'] = sent_topics_df.Vendors
        sent_topics_df['Total_sales'] = sent_topics_df.Total_sales
        sent_topics_df['Index'] = sent_topics_df.index
        sent_topics_df['Dominant_Topic'] = sent_topics_df.Data.apply(lambda x: x[0])
        sent_topics_df['Perc_Contribution'] = sent_topics_df.Data.apply(lambda x: round(x[1],4))
        sent_topics_df['Topic_Keywords'] = sent_topics_df.Dominant_Topic.apply(lambda x: top10dic

        sent_topics_df = sent_topics_df[['Dominant_Topic', 'Perc_Contribution', 'Topic_Keywords',
        return(sent_topics_df)

    vendor = df2[other]['vendor']
    sales = df2[other]['total_sales']
    df2_topic_sents_keywords_other = format_topics_sentences(ldamodel=optimal_model_other, corpus

    # Format
    df2_dominant_topic_other = df2_topic_sents_keywords_other.reset_index()
    df2_dominant_topic_other.columns = ['Document_No', 'Dominant_Topic', 'Topic_Perc_Contrib', 'K
```

In [101]:
```
    # Group top 5 sentences under each topic
    sent_topics_sorted_other = pd.DataFrame()

    sent_topics_outdf_grpd = df2_topic_sents_keywords_other.groupby('Dominant_Topic')

    for i, grp in sent_topics_outdf_grpd:
        sent_topics_sorted_other = pd.concat([sent_topics_sorted_other,
                                              grp.sort_values(['Perc_Contribution'], ascending
                                             axis=0)


    # Reset Index
    sent_topics_sorted_other.reset_index(drop=True, inplace=True)

    # Format
    sent_topics_sorted_other.columns = ['Topic_Num', "Topic_Perc_Contrib", "Keywords", "Text", "I
```

In [102]:
```
    cluster0 = df2_dominant_topic_other.loc[df2_dominant_topic_other['Dominant_Topic'] == 0]
    cluster1 = df2_dominant_topic_other.loc[df2_dominant_topic_other['Dominant_Topic'] == 1]
    cluster2 = df2_dominant_topic_other.loc[df2_dominant_topic_other['Dominant_Topic'] == 2]
    cluster3 = df2_dominant_topic_other.loc[df2_dominant_topic_other['Dominant_Topic'] == 3]

    d = {'Cluster_No' : pd.Series([0,1,2,3]),
```

```
            'Total_listings' : pd.Series([len(df2_dominant_topic_other)]),
            'Numb_of_list ' : pd.Series([len(cluster0), len(cluster1), len(cluster2), len(clus
            'Total_uniq_vendors' : pd.Series([df2_dominant_topic_other.Vendors.nunique()]),
            'Unique_vendors' : pd.Series([cluster0.Vendors.nunique(),cluster1.Vendors.nunique(
            'Total_revenue' : pd.Series([df2_dominant_topic_other.Total_sales.sum()]),
            'Revenue' : pd.Series([cluster0.Total_sales.sum(),cluster1.Total_sales.sum(),clust
        df_intel = pd.DataFrame(d)
        cols = df_intel.columns.tolist()
        cols = ['Cluster_No',
         'Total_listings',
         'Numb_of_list ',
         'Total_uniq_vendors',
         'Unique_vendors',
         'Total_revenue',
         'Revenue']
        df_intel = df_intel[cols]
```

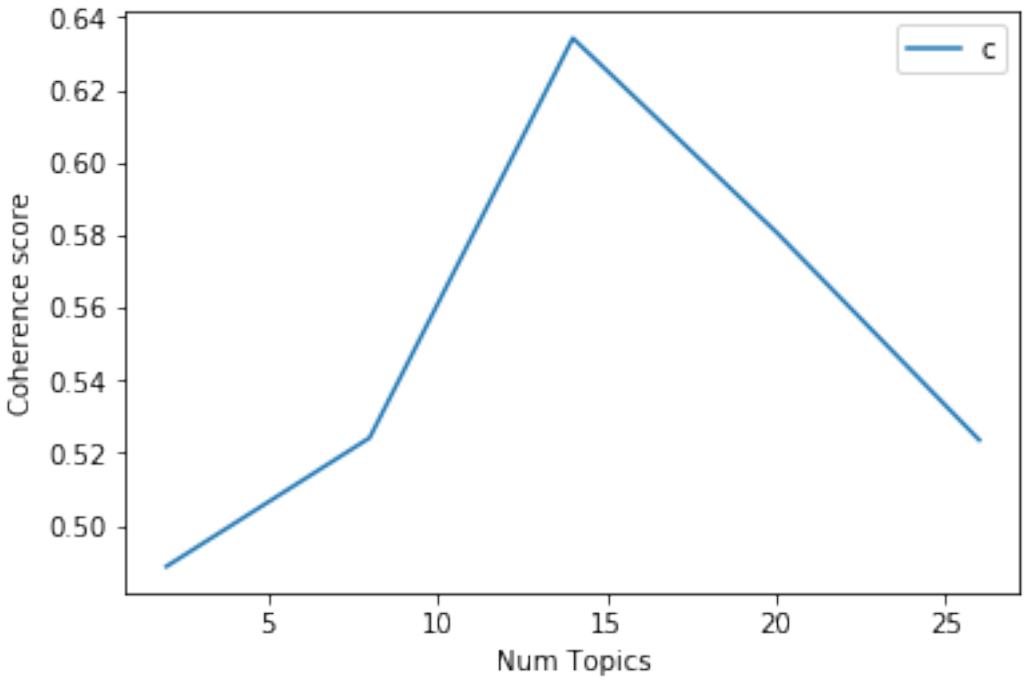
```
In [103]: writer = ExcelWriter('D:/fegmiedema/Desktop/Other.xlsx')
          sent_topics_sorted_other.to_excel(writer,'Sheet1',index=False)
          df_intel.to_excel(writer,'Sheet2',index=False)
          writer.save()
```

B.2.10. Account - 3804 listings - 14 topics

```
In [104]: data_lemmatized_other_account = lemmatization(df2[other_account]['title&description'])
          id2word_other_account = corpora.Dictionary(data_lemmatized_other_account)
          texts_other_account = data_lemmatized_other_account
          corpus_other_account = [id2word_other_account.doc2bow(text) for text in texts_other_acc
```

```
In [105]: model_list_other_account, coherence_values_other_account = compute_coherence_values(dic
```

```
In [106]: # Show graph
          limit=30; start=2; step=6;
          x = range(start, limit, step)
          plt.plot(x, coherence_values_other_account)
          plt.xlabel("Num Topics")
          plt.ylabel("Coherence score")
          plt.legend(("coherence_values"), loc='best')
          plt.show()
```




```
In [107]: # Print the coherence scores
          for m, cv in zip(x, coherence_values_other_account):
              print("Num Topics =", m, " has Coherence Value of", round(cv, 4))

Num Topics = 2  has Coherence Value of 0.4888
Num Topics = 8  has Coherence Value of 0.5242
Num Topics = 14  has Coherence Value of 0.6343
Num Topics = 20  has Coherence Value of 0.5809
Num Topics = 26  has Coherence Value of 0.5236
```

```
In [110]: optimal_model_other_account = model_list_other_account[2]
          vis = pyLDAvis.gensim.prepare(optimal_model_other_account, corpus_other_account, id2word_othe
          pyLDAvis.display(vis)

Out[110]: <IPython.core.display.HTML object>
```

```
In [111]: def format_topics_sentences(ldamodel, corpus, texts, original, vendor, sales):
              # Array of top 10 topics
              top10array = []

              for row in range(ldamodel.num_topics):
                  wp = ldamodel.show_topic(row)
                  topic_keywords = ", ".join([word for word, prop in wp])
                  top10array.append((row, topic_keywords))

              top10dict = dict(top10array)

              sent_topics_df = pd.DataFrame(pd.DataFrame([sorted(topic[0], key=lambda x: (x[1]), revers
              sent_topics_df.columns=["Data"]
```

```
        original = original.reset_index()
        sent_topics_df = pd.concat([sent_topics_df, original], axis=1)
        sent_topics_df.columns.values[2] = 'Original'

        vendors = vendor.reset_index()
        sent_topics_df = pd.concat([sent_topics_df, vendors], axis=1)
        sent_topics_df.columns.values[4] = 'Vendors'

        total_sales = sales.reset_index()
        sent_topics_df = pd.concat([sent_topics_df, total_sales], axis=1)
        sent_topics_df.columns.values[5] = 'Total_sales'

        sent_topics_df['Original'] = sent_topics_df.Original
        sent_topics_df['Vendors'] = sent_topics_df.Vendors
        sent_topics_df['Total_sales'] = sent_topics_df.Total_sales
        sent_topics_df['Index'] = sent_topics_df.index
        sent_topics_df['Dominant_Topic'] = sent_topics_df.Data.apply(lambda x: x[0])
        sent_topics_df['Perc_Contribution'] = sent_topics_df.Data.apply(lambda x: round(x[1
        sent_topics_df['Topic_Keywords'] = sent_topics_df.Dominant_Topic.apply(lambda x: to

        sent_topics_df = sent_topics_df[['Dominant_Topic', 'Perc_Contribution', 'Topic_Keyw
        return(sent_topics_df)

    vendor = df2[other_account]['vendor']
    sales = df2[other_account]['total_sales']
    df2_topic_sents_keywords_other_account = format_topics_sentences(ldamodel=optimal_model

    # Format
    df2_dominant_topic_other_account = df2_topic_sents_keywords_other_account.reset_index()
    df2_dominant_topic_other_account.columns = ['Document_No', 'Dominant_Topic', 'Topic_Per
```

In [112]:
```
    # Group top 5 sentences under each topic
    sent_topics_sorted_other_account = pd.DataFrame()

    sent_topics_outdf_grpd = df2_topic_sents_keywords_other_account.groupby('Dominant_Topic

    for i, grp in sent_topics_outdf_grpd:
        sent_topics_sorted_other_account = pd.concat([sent_topics_sorted_other_account,
                                                 grp.sort_values(['Perc_Contribution'], asc
                                                axis=0)

    # Reset Index
    sent_topics_sorted_other_account.reset_index(drop=True, inplace=True)

    # Format
    sent_topics_sorted_other_account.columns = ['Topic_Num', "Topic_Perc_Contrib", "Keyword
```

In [116]:
```
    cluster0 = df2_dominant_topic_other_account.loc[df2_dominant_topic_other_account['Domin
    cluster1 = df2_dominant_topic_other_account.loc[df2_dominant_topic_other_account['Domin
    cluster2 = df2_dominant_topic_other_account.loc[df2_dominant_topic_other_account['Domin
    cluster3 = df2_dominant_topic_other_account.loc[df2_dominant_topic_other_account['Domin
    cluster4 = df2_dominant_topic_other_account.loc[df2_dominant_topic_other_account['Domin
    cluster5 = df2_dominant_topic_other_account.loc[df2_dominant_topic_other_account['Domin
    cluster6 = df2_dominant_topic_other_account.loc[df2_dominant_topic_other_account['Domin
    cluster7 = df2_dominant_topic_other_account.loc[df2_dominant_topic_other_account['Domin
    cluster8 = df2_dominant_topic_other_account.loc[df2_dominant_topic_other_account['Domin
    cluster9 = df2_dominant_topic_other_account.loc[df2_dominant_topic_other_account['Domin
```

```
cluster10 = df2_dominant_topic_other_account.loc[df2_dominant_topic_other_account['Dominant_T
cluster11 = df2_dominant_topic_other_account.loc[df2_dominant_topic_other_account['Dominant_T
cluster12 = df2_dominant_topic_other_account.loc[df2_dominant_topic_other_account['Dominant_T
cluster13 = df2_dominant_topic_other_account.loc[df2_dominant_topic_other_account['Dominant_T

d = {'Cluster_No' : pd.Series([0,1,2,3,4,5,6,7,8,9,10,11,12,13]),
     'Total_listings' : pd.Series([len(df2_dominant_topic_other_account)]),
     'Numb_of_list ' : pd.Series([len(cluster0), len(cluster1), len(cluster2), len(cluster3),
     'Total_uniq_vendors' : pd.Series([df2_dominant_topic_other_account.Vendors.nunique()]),
     'Unique_vendors' : pd.Series([cluster0.Vendors.nunique(),cluster1.Vendors.nunique(),clus
     'Total_revenue' : pd.Series([df2_dominant_topic_other_account.Total_sales.sum()]),
     'Revenue' : pd.Series([cluster0.Total_sales.sum(),cluster1.Total_sales.sum(),cluster2.To
df_intel = pd.DataFrame(d)
cols = df_intel.columns.tolist()
cols = ['Cluster_No',
 'Total_listings',
 'Numb_of_list ',
 'Total_uniq_vendors',
 'Unique_vendors',
 'Total_revenue',
 'Revenue']
df_intel = df_intel[cols]
```

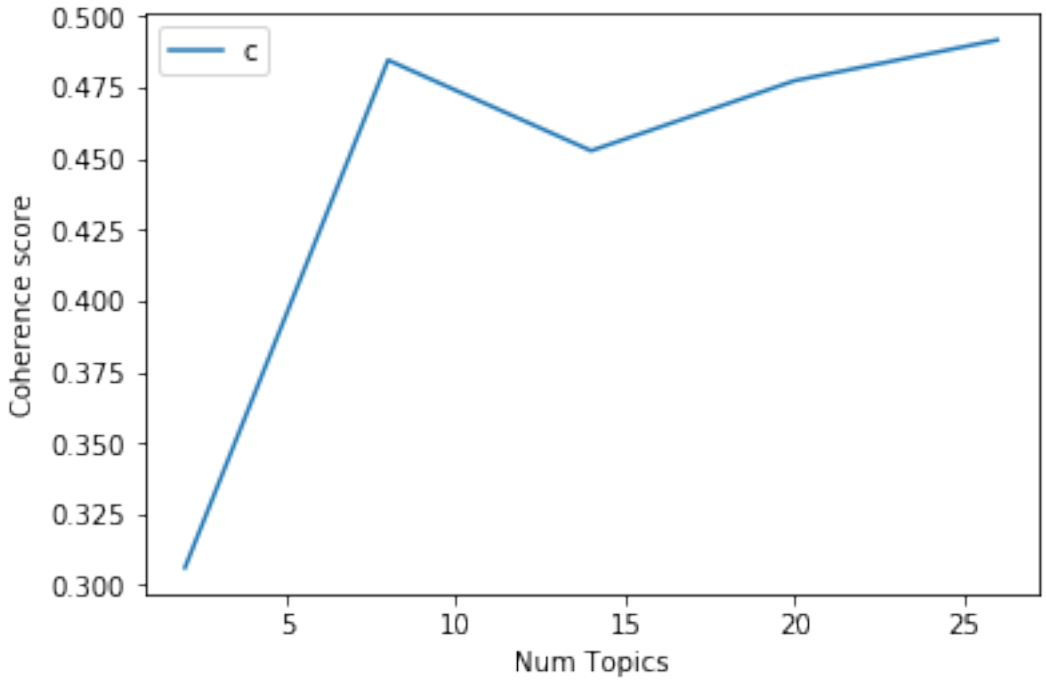
```
In [117]: writer = ExcelWriter('D:/fegmiedema/Desktop/Other_account.xlsx')
          sent_topics_sorted_other_account.to_excel(writer,'Sheet1',index=False)
          df_intel.to_excel(writer,'Sheet2',index=False)
          writer.save()
```

B.2.11. Custom - 6376 listings - 18 topics

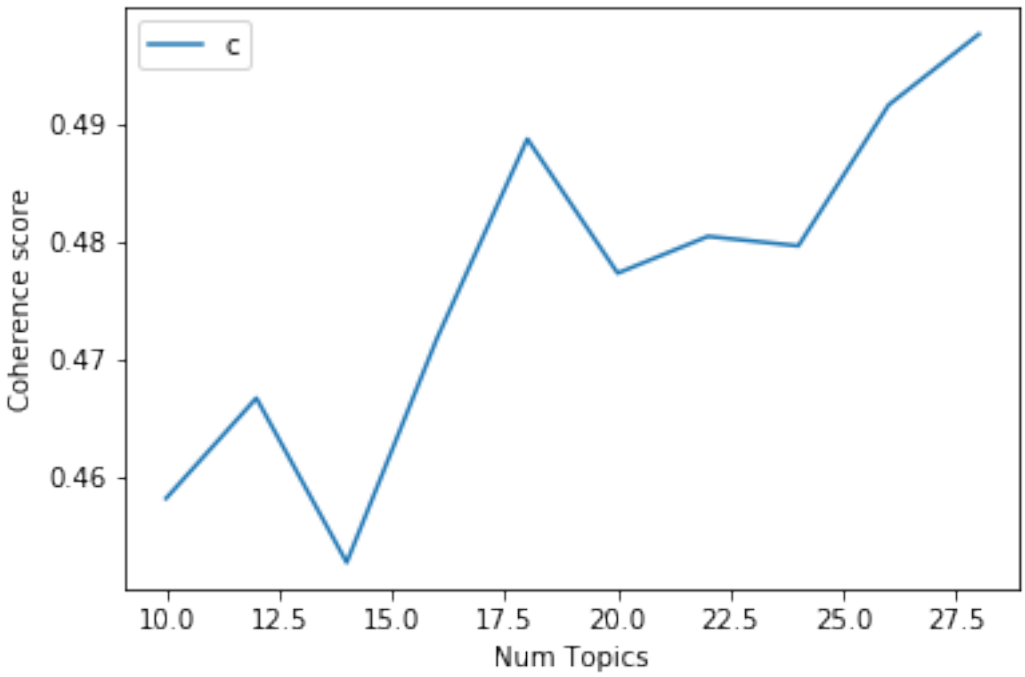
```
In [118]: data_lemmatized_other_custom = lemmatization(df2[other_custom]['title&description'])
          id2word_other_custom = corpora.Dictionary(data_lemmatized_other_custom)
          texts_other_custom = data_lemmatized_other_custom
          corpus_other_custom = [id2word_other_custom.doc2bow(text) for text in texts_other_custom]
```

```
In [119]: model_list_other_custom, coherence_values_other_custom = compute_coherence_values(dictionary=
```

```
In [120]: # Show graph
          limit=30; start=2; step=6;
          x = range(start, limit, step)
          plt.plot(x, coherence_values_other_custom)
          plt.xlabel("Num Topics")
          plt.ylabel("Coherence score")
          plt.legend(("coherence_values"), loc='best')
          plt.show()
```




```
In [121]: # Print the coherence scores
          for m, cv in zip(x, coherence_values_other_custom):
              print("Num Topics =", m, " has Coherence Value of", round(cv, 4))

Num Topics = 2  has Coherence Value of 0.3062
Num Topics = 8  has Coherence Value of 0.4845
Num Topics = 14  has Coherence Value of 0.4527
Num Topics = 20  has Coherence Value of 0.4773
Num Topics = 26  has Coherence Value of 0.4916


In [123]: optimal_model_other_custom = model_list_other_custom[4]
          vis = pyLDAvis.gensim.prepare(optimal_model_other_custom, corpus_other_account, id2word
          pyLDAvis.display(vis)

Out[123]: <IPython.core.display.HTML object>

In [124]: model_list_other_custom2, coherence_values_other_custom2 = compute_coherence_values(dic

In [125]: # Show graph
          limit=30; start=10; step=2;
          x = range(start, limit, step)
          plt.plot(x, coherence_values_other_custom2)
          plt.xlabel("Num Topics")
          plt.ylabel("Coherence score")
          plt.legend(("coherence_values"), loc='best')
          plt.show()
```




```
In [126]: # Print the coherence scores
          for m, cv in zip(x, coherence_values_other_custom2):
              print("Num Topics =", m, " has Coherence Value of", round(cv, 4))

Num Topics = 10  has Coherence Value of 0.4581
Num Topics = 12  has Coherence Value of 0.4666
Num Topics = 14  has Coherence Value of 0.4527
Num Topics = 16  has Coherence Value of 0.4717
Num Topics = 18  has Coherence Value of 0.4887
Num Topics = 20  has Coherence Value of 0.4773
Num Topics = 22  has Coherence Value of 0.4804
Num Topics = 24  has Coherence Value of 0.4796
Num Topics = 26  has Coherence Value of 0.4916
Num Topics = 28  has Coherence Value of 0.4976
```

```
In [127]: optimal_model_other_custom = model_list_other_custom2[4]
          vis = pyLDAvis.gensim.prepare(optimal_model_other_custom, corpus_other_account, id2word_other
          pyLDAvis.display(vis)

Out[127]: <IPython.core.display.HTML object>
```

```
In [128]: def format_topics_sentences(ldamodel, corpus, texts, original, vendor, sales):
              # Array of top 10 topics
              top10array = []

              for row in range(ldamodel.num_topics):
                  wp = ldamodel.show_topic(row)
                  topic_keywords = ", ".join([word for word, prop in wp])
                  top10array.append((row, topic_keywords))
```

```
        top10dict = dict(top10array)

        sent_topics_df = pd.DataFrame(pd.DataFrame([sorted(topic[0], key=lambda x: (x[1]),
        sent_topics_df.columns=["Data"]

        original = original.reset_index()
        sent_topics_df = pd.concat([sent_topics_df, original], axis=1)
        sent_topics_df.columns.values[2] = 'Original'

        vendors = vendor.reset_index()
        sent_topics_df = pd.concat([sent_topics_df, vendors], axis=1)
        sent_topics_df.columns.values[4] = 'Vendors'

        total_sales = sales.reset_index()
        sent_topics_df = pd.concat([sent_topics_df, total_sales], axis=1)
        sent_topics_df.columns.values[5] = 'Total_sales'

        sent_topics_df['Original'] = sent_topics_df.Original
        sent_topics_df['Vendors'] = sent_topics_df.Vendors
        sent_topics_df['Total_sales'] = sent_topics_df.Total_sales
        sent_topics_df['Index'] = sent_topics_df.index
        sent_topics_df['Dominant_Topic'] = sent_topics_df.Data.apply(lambda x: x[0])
        sent_topics_df['Perc_Contribution'] = sent_topics_df.Data.apply(lambda x: round(x[1
        sent_topics_df['Topic_Keywords'] = sent_topics_df.Dominant_Topic.apply(lambda x: to

        sent_topics_df = sent_topics_df[['Dominant_Topic', 'Perc_Contribution', 'Topic_Keyw
        return(sent_topics_df)

    vendor = df2[other_custom]['vendor']
    sales = df2[other_custom]['total_sales']
    df2_topic_sents_keywords_other_custom = format_topics_sentences(ldamodel=optimal_model_

    # Format
    df2_dominant_topic_other_custom = df2_topic_sents_keywords_other_custom.reset_index()
    df2_dominant_topic_other_custom.columns = ['Document_No', 'Dominant_Topic', 'Topic_Perc
```

In [129]:
```
    # Group top 5 sentences under each topic
    sent_topics_sorted_other_custom = pd.DataFrame()

    sent_topics_outdf_grpd = df2_topic_sents_keywords_other_custom.groupby('Dominant_Topic'

    for i, grp in sent_topics_outdf_grpd:
        sent_topics_sorted_other_custom = pd.concat([sent_topics_sorted_other_custom,
                                                     grp.sort_values(['Perc_Contribution'], asc
                                                    axis=0)

    # Reset Index
    sent_topics_sorted_other_custom.reset_index(drop=True, inplace=True)

    # Format
    sent_topics_sorted_other_custom.columns = ['Topic_Num', "Topic_Perc_Contrib", "Keywords
```

In [131]:
```
    cluster0 = df2_dominant_topic_other_custom.loc[df2_dominant_topic_other_custom['Dominan
    cluster1 = df2_dominant_topic_other_custom.loc[df2_dominant_topic_other_custom['Dominan
    cluster2 = df2_dominant_topic_other_custom.loc[df2_dominant_topic_other_custom['Dominan
    cluster3 = df2_dominant_topic_other_custom.loc[df2_dominant_topic_other_custom['Dominan
    cluster4 = df2_dominant_topic_other_custom.loc[df2_dominant_topic_other_custom['Dominan
```

```
cluster5 = df2_dominant_topic_other_custom.loc[df2_dominant_topic_other_custom['Dominant_Topi
cluster6 = df2_dominant_topic_other_custom.loc[df2_dominant_topic_other_custom['Dominant_Topi
cluster7 = df2_dominant_topic_other_custom.loc[df2_dominant_topic_other_custom['Dominant_Topi
cluster8 = df2_dominant_topic_other_custom.loc[df2_dominant_topic_other_custom['Dominant_Topi
cluster9 = df2_dominant_topic_other_custom.loc[df2_dominant_topic_other_custom['Dominant_Topi
cluster10 = df2_dominant_topic_other_custom.loc[df2_dominant_topic_other_custom['Dominant_Top
cluster11 = df2_dominant_topic_other_custom.loc[df2_dominant_topic_other_custom['Dominant_Top
cluster12 = df2_dominant_topic_other_custom.loc[df2_dominant_topic_other_custom['Dominant_Top
cluster13 = df2_dominant_topic_other_custom.loc[df2_dominant_topic_other_custom['Dominant_Top
cluster14 = df2_dominant_topic_other_custom.loc[df2_dominant_topic_other_custom['Dominant_Top
cluster15 = df2_dominant_topic_other_custom.loc[df2_dominant_topic_other_custom['Dominant_Top
cluster16 = df2_dominant_topic_other_custom.loc[df2_dominant_topic_other_custom['Dominant_Top
cluster17 = df2_dominant_topic_other_custom.loc[df2_dominant_topic_other_custom['Dominant_Top

d = {'Cluster_No' : pd.Series([0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17]),
     'Total_listings' : pd.Series([len(df2_dominant_topic_other_custom)]),
     'Numb_of_list ' : pd.Series([len(cluster0), len(cluster1), len(cluster2), len(cluster3),
     'Total_uniq_vendors' : pd.Series([df2_dominant_topic_other_custom.Vendors.nunique()]),
     'Unique_vendors' : pd.Series([cluster0.Vendors.nunique(),cluster1.Vendors.nunique(),clus
     'Total_revenue' : pd.Series([df2_dominant_topic_other_custom.Total_sales.sum()]),
     'Revenue' : pd.Series([cluster0.Total_sales.sum(),cluster1.Total_sales.sum(),cluster2.To
df_intel = pd.DataFrame(d)
cols = df_intel.columns.tolist()
cols = ['Cluster_No',
 'Total_listings',
 'Numb_of_list ',
 'Total_uniq_vendors',
 'Unique_vendors',
 'Total_revenue',
 'Revenue']
df_intel = df_intel[cols]
```

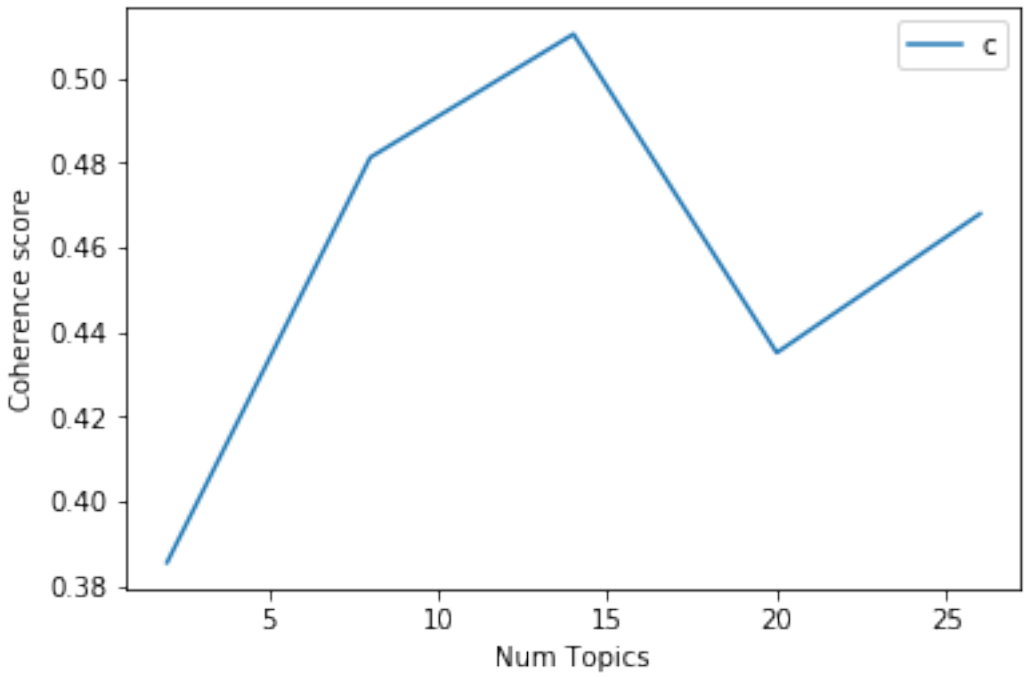
```
In [132]: writer = ExcelWriter('D:/fegmiedema/Desktop/Other_custom.xlsx')
          sent_topics_sorted_other_custom.to_excel(writer,'Sheet1',index=False)
          df_intel.to_excel(writer,'Sheet2',index=False)
          writer.save()
```

B.2.12. Fake - 3423 listings - 10 topics

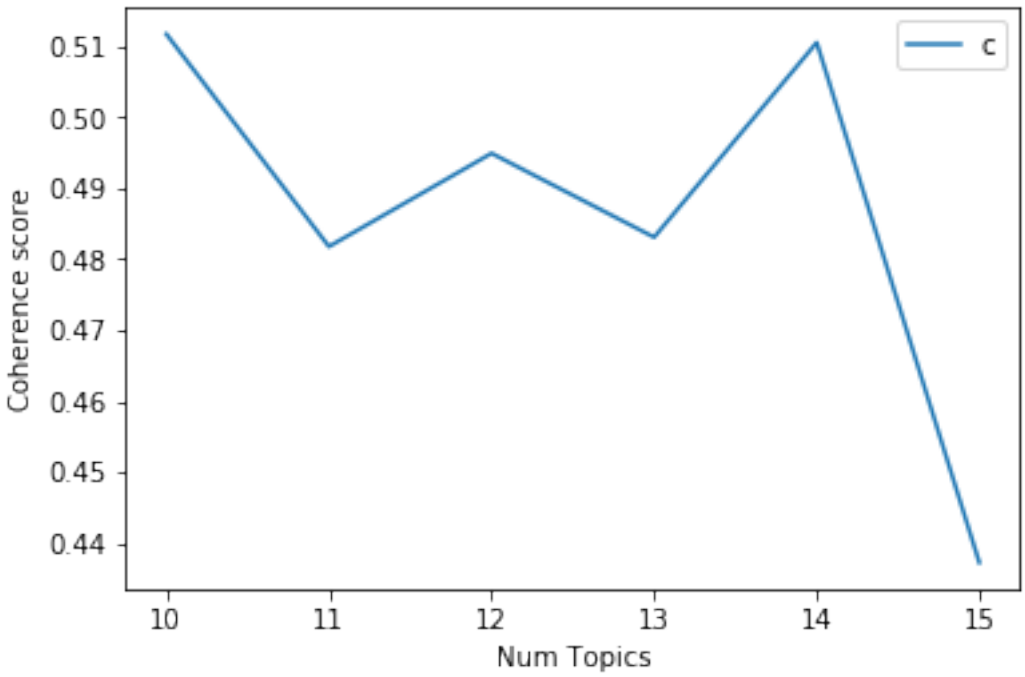
```
In [133]: data_lemmatized_other_fake = lemmatization(df2[other_fake]['title&description'])
          id2word_other_fake = corpora.Dictionary(data_lemmatized_other_fake)
          texts_other_fake = data_lemmatized_other_fake
          corpus_other_fake = [id2word_other_fake.doc2bow(text) for text in texts_other_fake]
```

```
In [134]: model_list_other_fake, coherence_values_other_fake = compute_coherence_values(dictionary=id2w
```

```
In [135]: # Show graph
          limit=30; start=2; step=6;
          x = range(start, limit, step)
          plt.plot(x, coherence_values_other_fake)
          plt.xlabel("Num Topics")
          plt.ylabel("Coherence score")
          plt.legend(("coherence_values"), loc='best')
          plt.show()
```




```
In [136]: # Print the coherence scores
          for m, cv in zip(x, coherence_values_other_fake):
              print("Num Topics =", m, " has Coherence Value of", round(cv, 4))

Num Topics = 2  has Coherence Value of 0.3854
Num Topics = 8  has Coherence Value of 0.4813
Num Topics = 14  has Coherence Value of 0.5105
Num Topics = 20  has Coherence Value of 0.4351
Num Topics = 26  has Coherence Value of 0.4679


In [137]: optimal_model_other_fake = model_list_other_fake[2]
          vis = pyLDAvis.gensim.prepare(optimal_model_other_fake, corpus_other_fake, id2word_othe
          pyLDAvis.display(vis)

Out[137]: <IPython.core.display.HTML object>

In [138]: model_list_other_fake2, coherence_values_other_fake2 = compute_coherence_values(diction

In [139]: # Show graph
          limit=16; start=10; step=1;
          x = range(start, limit, step)
          plt.plot(x, coherence_values_other_fake2)
          plt.xlabel("Num Topics")
          plt.ylabel("Coherence score")
          plt.legend(("coherence_values"), loc='best')
          plt.show()
```




```
In [140]: # Print the coherence scores
          for m, cv in zip(x, coherence_values_other_fake2):
              print("Num Topics =", m, " has Coherence Value of", round(cv, 4))

Num Topics = 10  has Coherence Value of 0.5117
Num Topics = 11  has Coherence Value of 0.4818
Num Topics = 12  has Coherence Value of 0.4949
Num Topics = 13  has Coherence Value of 0.4831
Num Topics = 14  has Coherence Value of 0.5105
Num Topics = 15  has Coherence Value of 0.4373
```

```
In [141]: optimal_model_other_fake = model_list_other_fake2[1]
          vis = pyLDAvis.gensim.prepare(optimal_model_other_fake, corpus_other_fake, id2word_other_fake
          pyLDAvis.display(vis)

Out[141]: <IPython.core.display.HTML object>
```

```
In [142]: def format_topics_sentences(ldamodel, corpus, texts, original, vendor, sales):
              # Array of top 10 topics
              top10array = []

              for row in range(ldamodel.num_topics):
                  wp = ldamodel.show_topic(row)
                  topic_keywords = ", ".join([word for word, prop in wp])
                  top10array.append((row, topic_keywords))

              top10dict = dict(top10array)

              sent_topics_df = pd.DataFrame(pd.DataFrame([sorted(topic[0], key=lambda x: (x[1]), revers
              sent_topics_df.columns=["Data"]
```

```
        original = original.reset_index()
        sent_topics_df = pd.concat([sent_topics_df, original], axis=1)
        sent_topics_df.columns.values[2] = 'Original'

        vendors = vendor.reset_index()
        sent_topics_df = pd.concat([sent_topics_df, vendors], axis=1)
        sent_topics_df.columns.values[4] = 'Vendors'

        total_sales = sales.reset_index()
        sent_topics_df = pd.concat([sent_topics_df, total_sales], axis=1)
        sent_topics_df.columns.values[5] = 'Total_sales'

        sent_topics_df['Original'] = sent_topics_df.Original
        sent_topics_df['Vendors'] = sent_topics_df.Vendors
        sent_topics_df['Total_sales'] = sent_topics_df.Total_sales
        sent_topics_df['Index'] = sent_topics_df.index
        sent_topics_df['Dominant_Topic'] = sent_topics_df.Data.apply(lambda x: x[0])
        sent_topics_df['Perc_Contribution'] = sent_topics_df.Data.apply(lambda x: round(x[1
        sent_topics_df['Topic_Keywords'] = sent_topics_df.Dominant_Topic.apply(lambda x: to

        sent_topics_df = sent_topics_df[['Dominant_Topic', 'Perc_Contribution', 'Topic_Keyw
        return(sent_topics_df)

    vendor = df2[other_fake]['vendor']
    sales = df2[other_fake]['total_sales']
    df2_topic_sents_keywords_other_fake = format_topics_sentences(ldamodel=optimal_model_ot

    # Format
    df2_dominant_topic_other_fake = df2_topic_sents_keywords_other_fake.reset_index()
    df2_dominant_topic_other_fake.columns = ['Document_No', 'Dominant_Topic', 'Topic_Perc_C
```

In [143]:
```
    # Group top 5 sentences under each topic
    sent_topics_sorted_other_fake = pd.DataFrame()

    sent_topics_outdf_grpd = df2_topic_sents_keywords_other_fake.groupby('Dominant_Topic')

    for i, grp in sent_topics_outdf_grpd:
        sent_topics_sorted_other_fake = pd.concat([sent_topics_sorted_other_fake,
                                                   grp.sort_values(['Perc_Contribution'], asc
                                                  axis=0)

    # Reset Index
    sent_topics_sorted_other_fake.reset_index(drop=True, inplace=True)

    # Format
    sent_topics_sorted_other_fake.columns = ['Topic_Num', "Topic_Perc_Contrib", "Keywords",
```

In [144]:
```
    cluster0 = df2_dominant_topic_other_fake.loc[df2_dominant_topic_other_fake['Dominant_To
    cluster1 = df2_dominant_topic_other_fake.loc[df2_dominant_topic_other_fake['Dominant_To
    cluster2 = df2_dominant_topic_other_fake.loc[df2_dominant_topic_other_fake['Dominant_To
    cluster3 = df2_dominant_topic_other_fake.loc[df2_dominant_topic_other_fake['Dominant_To
    cluster4 = df2_dominant_topic_other_fake.loc[df2_dominant_topic_other_fake['Dominant_To
    cluster5 = df2_dominant_topic_other_fake.loc[df2_dominant_topic_other_fake['Dominant_To
    cluster6 = df2_dominant_topic_other_fake.loc[df2_dominant_topic_other_fake['Dominant_To
    cluster7 = df2_dominant_topic_other_fake.loc[df2_dominant_topic_other_fake['Dominant_To
    cluster8 = df2_dominant_topic_other_fake.loc[df2_dominant_topic_other_fake['Dominant_To
```

```
cluster9 = df2_dominant_topic_other_fake.loc[df2_dominant_topic_other_fake['Dominant_Topic']

d = {'Cluster_No' : pd.Series([0,1,2,3,4,5,6,7,8,9]),
     'Total_listings' : pd.Series([len(df2_dominant_topic_other_fake)]),
     'Numb_of_list ' : pd.Series([len(cluster0), len(cluster1), len(cluster2), len(cluster3),
     'Total_uniq_vendors' : pd.Series([df2_dominant_topic_other_fake.Vendors.nunique()]),
     'Unique_vendors' : pd.Series([cluster0.Vendors.nunique(),cluster1.Vendors.nunique(),clus
     'Total_revenue' : pd.Series([df2_dominant_topic_other_fake.Total_sales.sum()]),
     'Revenue' : pd.Series([cluster0.Total_sales.sum(),cluster1.Total_sales.sum(),cluster2.To
df_intel = pd.DataFrame(d)
cols = df_intel.columns.tolist()
cols = ['Cluster_No',
 'Total_listings',
 'Numb_of_list ',
 'Total_uniq_vendors',
 'Unique_vendors',
 'Total_revenue',
 'Revenue']
df_intel = df_intel[cols]
```

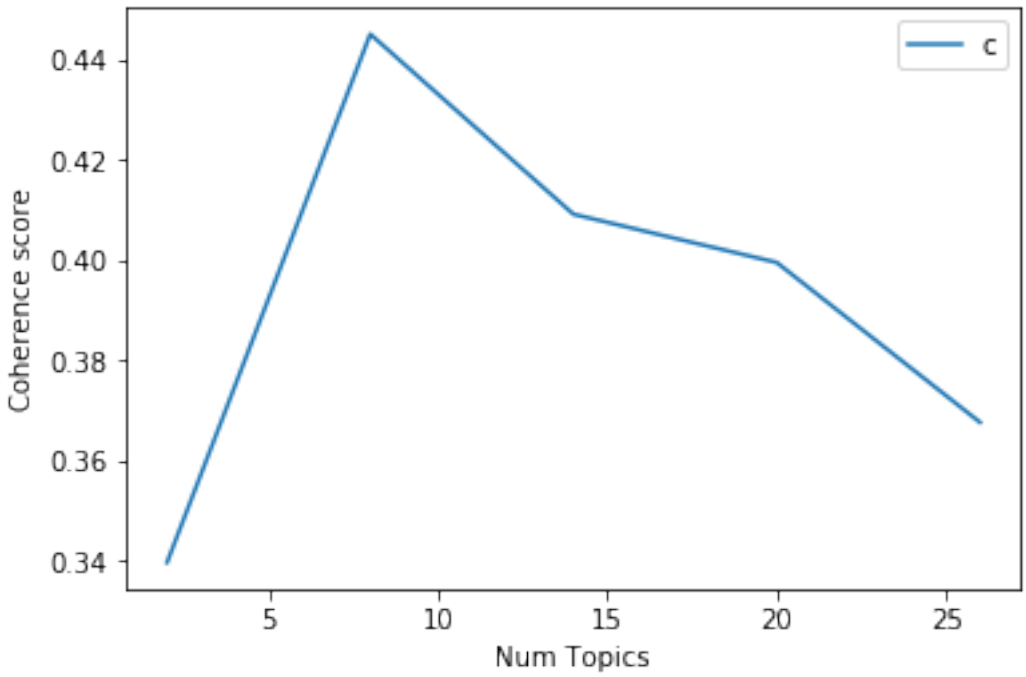
```
In [145]: writer = ExcelWriter('D:/fegmiedema/Desktop/Other_fake.xlsx')
          sent_topics_sorted_other_fake.to_excel(writer,'Sheet1',index=False)
          df_intel.to_excel(writer,'Sheet2',index=False)
          writer.save()
```

B.2.13. Guide - 5097 listings - 8 topics

```
In [147]: data_lemmatized_other_guide = lemmatization(df2[other_guide]['title&description'])
          id2word_other_guide = corpora.Dictionary(data_lemmatized_other_guide)
          texts_other_guide = data_lemmatized_other_guide
          corpus_other_guide  = [id2word_other_guide .doc2bow(text) for text in texts_other_guide ]
```

```
In [148]: model_list_other_guide, coherence_values_other_guide = compute_coherence_values(dictionary=id
```

```
In [149]: # Show graph
          limit=30; start=2; step=6;
          x = range(start, limit, step)
          plt.plot(x, coherence_values_other_guide)
          plt.xlabel("Num Topics")
          plt.ylabel("Coherence score")
          plt.legend(("coherence_values"), loc='best')
          plt.show()
```




```
In [150]: # Print the coherence scores
          for m, cv in zip(x, coherence_values_other_guide):
              print("Num Topics =", m, " has Coherence Value of", round(cv, 4))

Num Topics = 2  has Coherence Value of 0.3396
Num Topics = 8  has Coherence Value of 0.4451
Num Topics = 14  has Coherence Value of 0.4091
Num Topics = 20  has Coherence Value of 0.3995
Num Topics = 26  has Coherence Value of 0.3676
```

```
In [151]: optimal_model_other_guide = model_list_other_guide[1]
          vis = pyLDAvis.gensim.prepare(optimal_model_other_guide, corpus_other_guide, id2word_ot
          pyLDAvis.display(vis)

Out[151]: <IPython.core.display.HTML object>
```

```
In [157]: def format_topics_sentences(ldamodel, corpus, texts, original, vendor, sales):
              # Array of top 10 topics
              top10array = []

              for row in range(ldamodel.num_topics):
                  wp = ldamodel.show_topic(row)
                  topic_keywords = ", ".join([word for word, prop in wp])
                  top10array.append((row, topic_keywords))

              top10dict = dict(top10array)

              sent_topics_df = pd.DataFrame(pd.DataFrame([sorted(topic[0], key=lambda x: (x[1]),
              sent_topics_df.columns=["Data"]
```

```
        original = original.reset_index()
        sent_topics_df = pd.concat([sent_topics_df, original], axis=1)
        sent_topics_df.columns.values[2] = 'Original'

        vendors = vendor.reset_index()
        sent_topics_df = pd.concat([sent_topics_df, vendors], axis=1)
        sent_topics_df.columns.values[4] = 'Vendors'

        total_sales = sales.reset_index()
        sent_topics_df = pd.concat([sent_topics_df, total_sales], axis=1)
        sent_topics_df.columns.values[5] = 'Total_sales'

        sent_topics_df['Original'] = sent_topics_df.Original
        sent_topics_df['Vendors'] = sent_topics_df.Vendors
        sent_topics_df['Total_sales'] = sent_topics_df.Total_sales
        sent_topics_df['Index'] = sent_topics_df.index
        sent_topics_df['Dominant_Topic'] = sent_topics_df.Data.apply(lambda x: x[0])
        sent_topics_df['Perc_Contribution'] = sent_topics_df.Data.apply(lambda x: round(x[1],4))
        sent_topics_df['Topic_Keywords'] = sent_topics_df.Dominant_Topic.apply(lambda x: top10dic

        sent_topics_df = sent_topics_df[['Dominant_Topic', 'Perc_Contribution', 'Topic_Keywords',
        return(sent_topics_df)

vendor = df2[other_guide]['vendor']
sales = df2[other_guide]['total_sales']
df2_topic_sents_keywords_other_guide = format_topics_sentences(ldamodel=optimal_model_other_g

# Format
df2_dominant_topic_other_guide = df2_topic_sents_keywords_other_guide.reset_index()
df2_dominant_topic_other_guide.columns = ['Document_No', 'Dominant_Topic', 'Topic_Perc_Contri
```

In [158]:
```
# Group top 5 sentences under each topic
sent_topics_sorted_other_guide = pd.DataFrame()

sent_topics_outdf_grpd = df2_topic_sents_keywords_other_guide.groupby('Dominant_Topic')

for i, grp in sent_topics_outdf_grpd:
    sent_topics_sorted_other_guide = pd.concat([sent_topics_sorted_other_guide,
                                                grp.sort_values(['Perc_Contribution'], ascending
                                               axis=0)

# Reset Index
sent_topics_sorted_other_guide.reset_index(drop=True, inplace=True)

# Format
sent_topics_sorted_other_guide.columns = ['Topic_Num', "Topic_Perc_Contrib", "Keywords", "Tex
```

In [159]:
```
cluster0 = df2_dominant_topic_other_guide.loc[df2_dominant_topic_other_guide['Dominant_Topic'
cluster1 = df2_dominant_topic_other_guide.loc[df2_dominant_topic_other_guide['Dominant_Topic'
cluster2 = df2_dominant_topic_other_guide.loc[df2_dominant_topic_other_guide['Dominant_Topic'
cluster3 = df2_dominant_topic_other_guide.loc[df2_dominant_topic_other_guide['Dominant_Topic'
cluster4 = df2_dominant_topic_other_guide.loc[df2_dominant_topic_other_guide['Dominant_Topic'
cluster5 = df2_dominant_topic_other_guide.loc[df2_dominant_topic_other_guide['Dominant_Topic'
cluster6 = df2_dominant_topic_other_guide.loc[df2_dominant_topic_other_guide['Dominant_Topic'
cluster7 = df2_dominant_topic_other_guide.loc[df2_dominant_topic_other_guide['Dominant_Topic'

d = {'Cluster_No' : pd.Series([0,1,2,3,4,5,6,7]),
```

```
        'Total_listings' : pd.Series([len(df2_dominant_topic_other_guide)]),
        'Numb_of_list ' : pd.Series([len(cluster0), len(cluster1), len(cluster2), len(clus
        'Total_uniq_vendors' : pd.Series([df2_dominant_topic_other_guide.Vendors.nunique()
        'Unique_vendors' : pd.Series([cluster0.Vendors.nunique(),cluster1.Vendors.nunique(
        'Total_revenue' : pd.Series([df2_dominant_topic_other_guide.Total_sales.sum()]),
        'Revenue' : pd.Series([cluster0.Total_sales.sum(),cluster1.Total_sales.sum(),clust
df_intel = pd.DataFrame(d)
cols = df_intel.columns.tolist()
cols = ['Cluster_No',
 'Total_listings',
 'Numb_of_list ',
 'Total_uniq_vendors',
 'Unique_vendors',
 'Total_revenue',
 'Revenue']
df_intel = df_intel[cols]
```

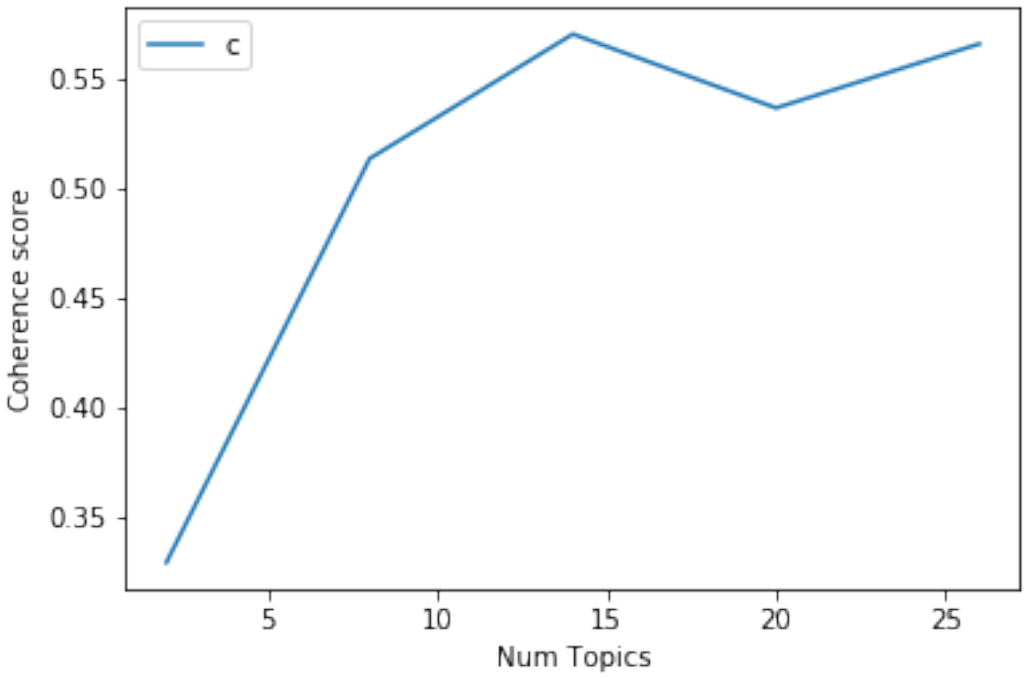
```
In [160]: writer = ExcelWriter('D:/fegmiedema/Desktop/Other_guide.xlsx')
          sent_topics_sorted_other_guide.to_excel(writer,'Sheet1',index=False)
          df_intel.to_excel(writer,'Sheet2',index=False)
          writer.save()
```

B.2.14. Pirated - 1429 listings - 12 topics

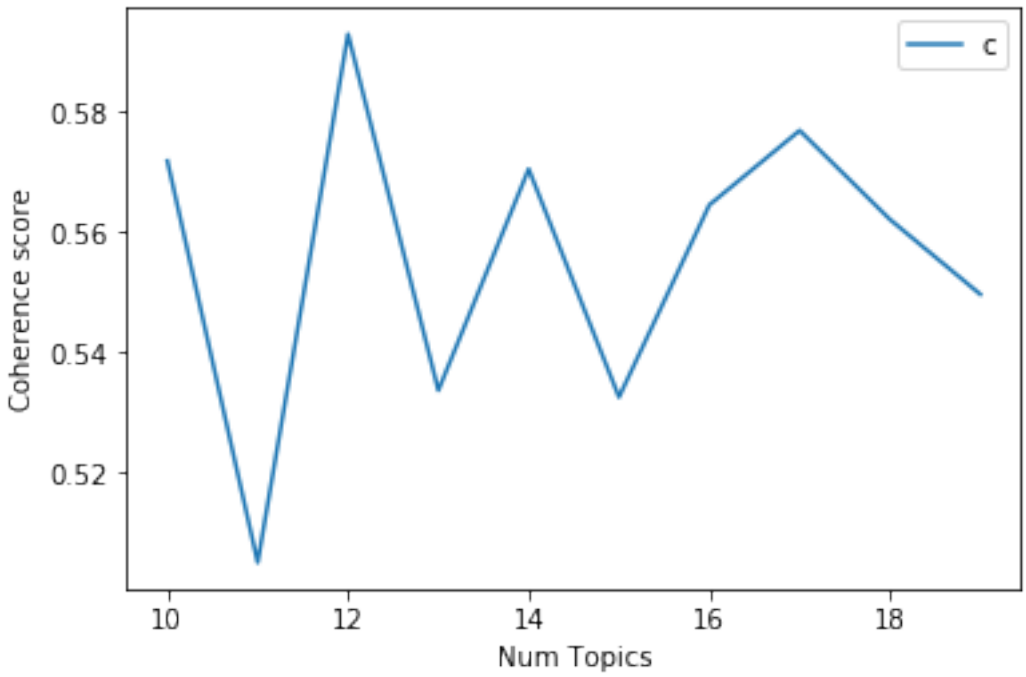
```
In [161]: data_lemmatized_other_pirated = lemmatization(df2[other_pirated]['title&description'])
          id2word_other_pirated = corpora.Dictionary(data_lemmatized_other_pirated)
          texts_other_pirated = data_lemmatized_other_pirated
          corpus_other_pirated = [id2word_other_pirated.doc2bow(text) for text in texts_other_pir
```

```
In [162]: model_list_other_pirated, coherence_values_other_pirated = compute_coherence_values(dic
```

```
In [163]: # Show graph
          limit=30; start=2; step=6;
          x = range(start, limit, step)
          plt.plot(x, coherence_values_other_pirated)
          plt.xlabel("Num Topics")
          plt.ylabel("Coherence score")
          plt.legend(("coherence_values"), loc='best')
          plt.show()
```




```
In [164]: # Print the coherence scores
          for m, cv in zip(x, coherence_values_other_pirated):
              print("Num Topics =", m, " has Coherence Value of", round(cv, 4))

Num Topics = 2  has Coherence Value of 0.3293
Num Topics = 8  has Coherence Value of 0.5136
Num Topics = 14  has Coherence Value of 0.5705
Num Topics = 20  has Coherence Value of 0.5368
Num Topics = 26  has Coherence Value of 0.566


In [166]: optimal_model_other_pirated = model_list_other_pirated[2]
          vis = pyLDAvis.gensim.prepare(optimal_model_other_pirated, corpus_other_pirated, id2word_othe
          pyLDAvis.display(vis)

Out[166]: <IPython.core.display.HTML object>

In [167]: model_list_other_pirated2, coherence_values_other_pirated2 = compute_coherence_values(diction

In [171]: # Show graph
          limit=20; start=10; step=1;
          x = range(start, limit, step)
          plt.plot(x, coherence_values_other_pirated2)
          plt.xlabel("Num Topics")
          plt.ylabel("Coherence score")
          plt.legend(("coherence_values"), loc='best')
          plt.show()
```




```
In [172]: # Print the coherence scores
          for m, cv in zip(x, coherence_values_other_pirated2):
              print("Num Topics =", m, " has Coherence Value of", round(cv, 4))

Num Topics = 10  has Coherence Value of 0.5718
Num Topics = 11  has Coherence Value of 0.5052
Num Topics = 12  has Coherence Value of 0.5929
Num Topics = 13  has Coherence Value of 0.5337
Num Topics = 14  has Coherence Value of 0.5705
Num Topics = 15  has Coherence Value of 0.5326
Num Topics = 16  has Coherence Value of 0.5645
Num Topics = 17  has Coherence Value of 0.5769
Num Topics = 18  has Coherence Value of 0.5621
Num Topics = 19  has Coherence Value of 0.5497
```

```
In [173]: optimal_model_other_pirated = model_list_other_pirated2[2]
          vis = pyLDAvis.gensim.prepare(optimal_model_other_pirated, corpus_other_pirated, id2wor
          pyLDAvis.display(vis)

Out[173]: <IPython.core.display.HTML object>
```

```
In [174]: def format_topics_sentences(ldamodel, corpus, texts, original, vendor, sales):
              # Array of top 10 topics
              top10array = []

              for row in range(ldamodel.num_topics):
                  wp = ldamodel.show_topic(row)
                  topic_keywords = ", ".join([word for word, prop in wp])
                  top10array.append((row, topic_keywords))
```

```
        top10dict = dict(top10array)

        sent_topics_df = pd.DataFrame(pd.DataFrame([sorted(topic[0], key=lambda x: (x[1]), revers
        sent_topics_df.columns=["Data"]

        original = original.reset_index()
        sent_topics_df = pd.concat([sent_topics_df, original], axis=1)
        sent_topics_df.columns.values[2] = 'Original'

        vendors = vendor.reset_index()
        sent_topics_df = pd.concat([sent_topics_df, vendors], axis=1)
        sent_topics_df.columns.values[4] = 'Vendors'

        total_sales = sales.reset_index()
        sent_topics_df = pd.concat([sent_topics_df, total_sales], axis=1)
        sent_topics_df.columns.values[5] = 'Total_sales'

        sent_topics_df['Original'] = sent_topics_df.Original
        sent_topics_df['Vendors'] = sent_topics_df.Vendors
        sent_topics_df['Total_sales'] = sent_topics_df.Total_sales
        sent_topics_df['Index'] = sent_topics_df.index
        sent_topics_df['Dominant_Topic'] = sent_topics_df.Data.apply(lambda x: x[0])
        sent_topics_df['Perc_Contribution'] = sent_topics_df.Data.apply(lambda x: round(x[1],4))
        sent_topics_df['Topic_Keywords'] = sent_topics_df.Dominant_Topic.apply(lambda x: top10dic

        sent_topics_df = sent_topics_df[['Dominant_Topic', 'Perc_Contribution', 'Topic_Keywords',
        return(sent_topics_df)

    vendor = df2[other_pirated]['vendor']
    sales = df2[other_pirated]['total_sales']
    df2_topic_sents_keywords_other_pirated = format_topics_sentences(ldamodel=optimal_model_other

    # Format
    df2_dominant_topic_other_pirated = df2_topic_sents_keywords_other_pirated.reset_index()
    df2_dominant_topic_other_pirated.columns = ['Document_No', 'Dominant_Topic', 'Topic_Perc_Cont
```

In [175]:
```
    # Group top 5 sentences under each topic
    sent_topics_sorted_other_pirated = pd.DataFrame()

    sent_topics_outdf_grpd = df2_topic_sents_keywords_other_pirated.groupby('Dominant_Topic')

    for i, grp in sent_topics_outdf_grpd:
        sent_topics_sorted_other_pirated = pd.concat([sent_topics_sorted_other_pirated,
                                                      grp.sort_values(['Perc_Contribution'], ascending
                                                     axis=0)

    # Reset Index
    sent_topics_sorted_other_pirated.reset_index(drop=True, inplace=True)

    # Format
    sent_topics_sorted_other_pirated.columns = ['Topic_Num', "Topic_Perc_Contrib", "Keywords", "T
```

In [176]:
```
    cluster0 = df2_dominant_topic_other_pirated.loc[df2_dominant_topic_other_pirated['Dominant_To
    cluster1 = df2_dominant_topic_other_pirated.loc[df2_dominant_topic_other_pirated['Dominant_To
    cluster2 = df2_dominant_topic_other_pirated.loc[df2_dominant_topic_other_pirated['Dominant_To
    cluster3 = df2_dominant_topic_other_pirated.loc[df2_dominant_topic_other_pirated['Dominant_To
    cluster4 = df2_dominant_topic_other_pirated.loc[df2_dominant_topic_other_pirated['Dominant_To
```

```
cluster5 = df2_dominant_topic_other_pirated.loc[df2_dominant_topic_other_pirated['Domin
cluster6 = df2_dominant_topic_other_pirated.loc[df2_dominant_topic_other_pirated['Domin
cluster7 = df2_dominant_topic_other_pirated.loc[df2_dominant_topic_other_pirated['Domin
cluster8 = df2_dominant_topic_other_pirated.loc[df2_dominant_topic_other_pirated['Domin
cluster9 = df2_dominant_topic_other_pirated.loc[df2_dominant_topic_other_pirated['Domin
cluster10 = df2_dominant_topic_other_pirated.loc[df2_dominant_topic_other_pirated['Domi
cluster11 = df2_dominant_topic_other_pirated.loc[df2_dominant_topic_other_pirated['Domi

d = {'Cluster_No' : pd.Series([0,1,2,3,4,5,6,7,8,9,10,11]),
     'Total_listings' : pd.Series([len(df2_dominant_topic_other_pirated)]),
     'Numb_of_list ' : pd.Series([len(cluster0), len(cluster1), len(cluster2), len(clus
     'Total_uniq_vendors' : pd.Series([df2_dominant_topic_other_pirated.Vendors.nunique
     'Unique_vendors' : pd.Series([cluster0.Vendors.nunique(),cluster1.Vendors.nunique(
     'Total_revenue' : pd.Series([df2_dominant_topic_other_pirated.Total_sales.sum()]),
     'Revenue' : pd.Series([cluster0.Total_sales.sum(),cluster1.Total_sales.sum(),clust
df_intel = pd.DataFrame(d)
cols = df_intel.columns.tolist()
cols = ['Cluster_No',
 'Total_listings',
 'Numb_of_list ',
 'Total_uniq_vendors',
 'Unique_vendors',
 'Total_revenue',
 'Revenue']
df_intel = df_intel[cols]
```

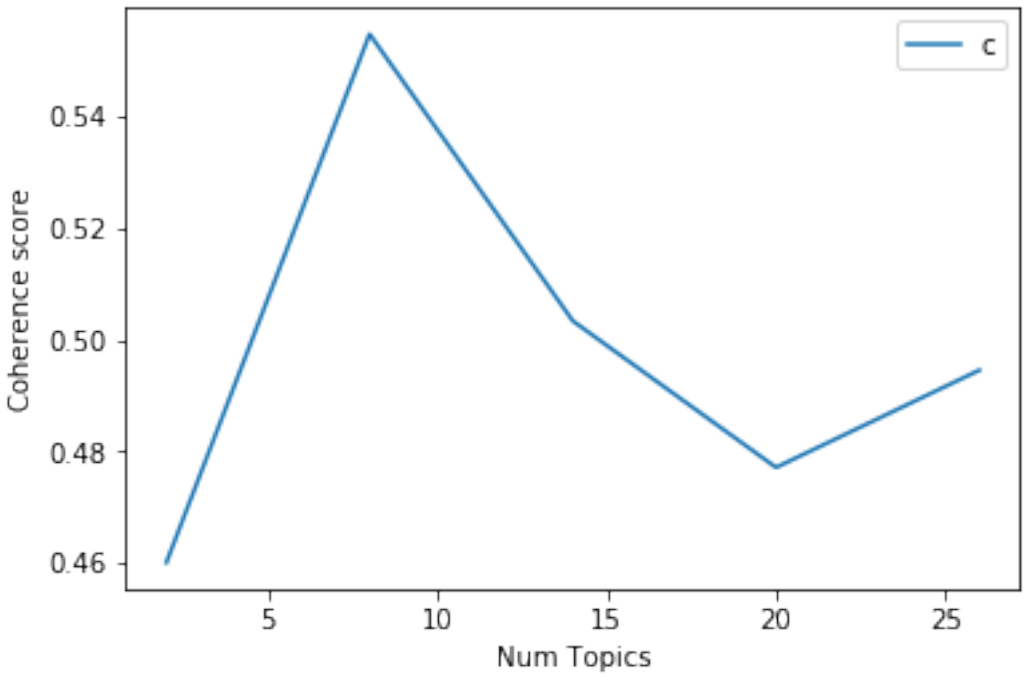
```
In [177]: writer = ExcelWriter('D:/fegmiedema/Desktop/Other_pirated.xlsx')
          sent_topics_sorted_other_pirated.to_excel(writer,'Sheet1',index=False)
          df_intel.to_excel(writer,'Sheet2',index=False)
          writer.save()
```

B.2.15. Voucher - 1300 listings - 8 topics

```
In [178]: data_lemmatized_other_voucher = lemmatization(df2[other_voucher]['title&description'])
          id2word_other_voucher = corpora.Dictionary(data_lemmatized_other_voucher)
          texts_other_voucher = data_lemmatized_other_voucher
          corpus_other_voucher = [id2word_other_voucher.doc2bow(text) for text in texts_other_vou
```

```
In [179]: model_list_other_voucher, coherence_values_other_voucher = compute_coherence_values(dic
```

```
In [180]: # Show graph
          limit=30; start=2; step=6;
          x = range(start, limit, step)
          plt.plot(x, coherence_values_other_voucher)
          plt.xlabel("Num Topics")
          plt.ylabel("Coherence score")
          plt.legend(("coherence_values"), loc='best')
          plt.show()
```




```
In [181]: # Print the coherence scores
          for m, cv in zip(x, coherence_values_other_voucher):
              print("Num Topics =", m, " has Coherence Value of", round(cv, 4))

Num Topics = 2  has Coherence Value of 0.4601
Num Topics = 8  has Coherence Value of 0.5548
Num Topics = 14  has Coherence Value of 0.5034
Num Topics = 20  has Coherence Value of 0.4771
Num Topics = 26  has Coherence Value of 0.4946
```

```
In [183]: optimal_model_other_voucher = model_list_other_voucher[1]
          vis = pyLDAvis.gensim.prepare(optimal_model_other_voucher, corpus_other_voucher, id2word_othe
          pyLDAvis.display(vis)

Out[183]: <IPython.core.display.HTML object>
```

```
In [191]: def format_topics_sentences(ldamodel, corpus, texts, original, vendor, sales, dutch):
              # Array of top 10 topics
              top10array = []

              for row in range(ldamodel.num_topics):
                  wp = ldamodel.show_topic(row)
                  topic_keywords = ", ".join([word for word, prop in wp])
                  top10array.append((row, topic_keywords))

              top10dict = dict(top10array)

              sent_topics_df = pd.DataFrame(pd.DataFrame([sorted(topic[0], key=lambda x: (x[1]), revers
              sent_topics_df.columns=["Data"]
```

```
            original = original.reset_index()
            sent_topics_df = pd.concat([sent_topics_df, original], axis=1)
            sent_topics_df.columns.values[2] = 'Original'

            vendors = vendor.reset_index()
            sent_topics_df = pd.concat([sent_topics_df, vendors], axis=1)
            sent_topics_df.columns.values[4] = 'Vendors'

            total_sales = sales.reset_index()
            sent_topics_df = pd.concat([sent_topics_df, total_sales], axis=1)
            sent_topics_df.columns.values[6] = 'Total_sales'

            dutch_id = dutch.reset_index()
            sent_topics_df = pd.concat([sent_topics_df, dutch_id], axis=1)
            sent_topics_df.columns.values[8] = 'Dutch_id'

            sent_topics_df['Original'] = sent_topics_df.Original
            sent_topics_df['Vendors'] = sent_topics_df.Vendors
            sent_topics_df['Total_sales'] = sent_topics_df.Total_sales
            sent_topics_df['Dutch_id'] = sent_topics_df.Dutch_id
            sent_topics_df['Index'] = sent_topics_df.index
            sent_topics_df['Dominant_Topic'] = sent_topics_df.Data.apply(lambda x: x[0])
            sent_topics_df['Perc_Contribution'] = sent_topics_df.Data.apply(lambda x: round(x[1
            sent_topics_df['Topic_Keywords'] = sent_topics_df.Dominant_Topic.apply(lambda x: to

            sent_topics_df = sent_topics_df[['Dominant_Topic', 'Perc_Contribution', 'Topic_Keyw
            return(sent_topics_df)

        vendor = df2[other_voucher]['vendor']
        sales = df2[other_voucher]['total_sales']
        dutch = df2[other_voucher]['dutch']
        df2_topic_sents_keywords_other_voucher = format_topics_sentences(ldamodel=optimal_model

        # Format
        df2_dominant_topic_other_voucher = df2_topic_sents_keywords_other_voucher.reset_index()
        df2_dominant_topic_other_voucher.columns = ['Document_No', 'Dominant_Topic', 'Topic_Per
```

In [192]:
```
        # Group top 5 sentences under each topic
        sent_topics_sorted_other_voucher = pd.DataFrame()

        sent_topics_outdf_grpd = df2_topic_sents_keywords_other_voucher.groupby('Dominant_Topic

        for i, grp in sent_topics_outdf_grpd:
            sent_topics_sorted_other_voucher = pd.concat([sent_topics_sorted_other_voucher,
                                                    grp.sort_values(['Perc_Contribution'], asc
                                                   axis=0)

        # Reset Index
        sent_topics_sorted_other_voucher.reset_index(drop=True, inplace=True)

        # Format
        sent_topics_sorted_other_voucher.columns = ['Topic_Num', "Topic_Perc_Contrib", "Keyword
```

In [193]:
```
        cluster0 = df2_dominant_topic_other_voucher.loc[df2_dominant_topic_other_voucher['Domin
        cluster1 = df2_dominant_topic_other_voucher.loc[df2_dominant_topic_other_voucher['Domin
        cluster2 = df2_dominant_topic_other_voucher.loc[df2_dominant_topic_other_voucher['Domin
        cluster3 = df2_dominant_topic_other_voucher.loc[df2_dominant_topic_other_voucher['Domin
```

```
cluster4 = df2_dominant_topic_other_voucher.loc[df2_dominant_topic_other_voucher['Dominant_To
cluster5 = df2_dominant_topic_other_voucher.loc[df2_dominant_topic_other_voucher['Dominant_To
cluster6 = df2_dominant_topic_other_voucher.loc[df2_dominant_topic_other_voucher['Dominant_To
cluster7 = df2_dominant_topic_other_voucher.loc[df2_dominant_topic_other_voucher['Dominant_To
cluster8 = df2_dominant_topic_other_voucher.loc[df2_dominant_topic_other_voucher['Dominant_To

d = {'Cluster_No' : pd.Series([0,1,2,3,4,5,6,7,8]),
     'Total_listings' : pd.Series([len(df2_dominant_topic_other_voucher)]),
     'Numb_of_list ' : pd.Series([len(cluster0), len(cluster1), len(cluster2), len(cluster3),
     'Total_uniq_vendors' : pd.Series([df2_dominant_topic_other_voucher.Vendors.nunique()]),
     'Unique_vendors' : pd.Series([cluster0.Vendors.nunique(),cluster1.Vendors.nunique(),clus
     'Total_revenue' : pd.Series([df2_dominant_topic_other_voucher.Total_sales.sum()]),
     'Revenue' : pd.Series([cluster0.Total_sales.sum(),cluster1.Total_sales.sum(),cluster2.To
    'Dutch_identifier' : pd.Series([cluster0.Dutch_id.sum(),cluster1.Dutch_id.sum(),cluster2.
df_intel = pd.DataFrame(d)
cols = df_intel.columns.tolist()
cols = ['Cluster_No',
 'Total_listings',
 'Numb_of_list ',
 'Total_uniq_vendors',
 'Unique_vendors',
 'Total_revenue',
 'Revenue',
 'Dutch_identifier']
df_intel = df_intel[cols]
```

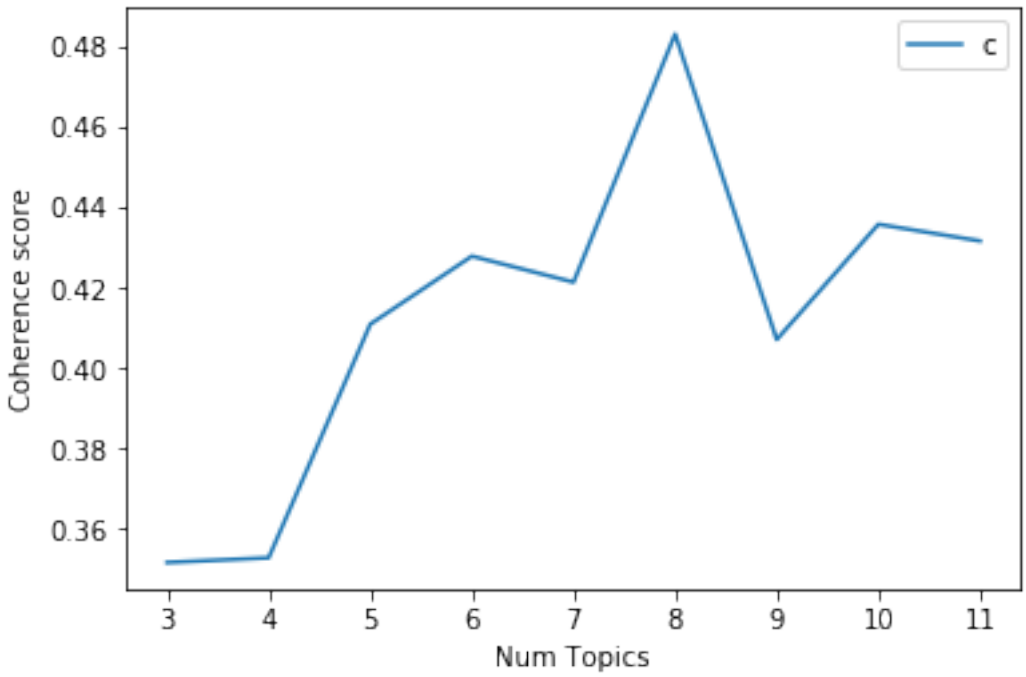
```
In [195]: writer = ExcelWriter('D:/fegmiedema/Desktop/Voucher.xlsx')
          sent_topics_sorted_other_voucher.to_excel(writer,'Sheet1',index=False)
          df_intel.to_excel(writer,'Sheet2',index=False)
          writer.save()
```

B.2.16. Phone - 267 listings - 8 topics

```
In [197]: data_lemmatized_phone = lemmatization(df2[phone]['title&description'])
          id2word_phone = corpora.Dictionary(data_lemmatized_phone)
          texts_phone = data_lemmatized_phone
          corpus_phone = [id2word_phone.doc2bow(text) for text in texts_phone]
```

```
In [207]: model_list_phone, coherence_values_phone= compute_coherence_values(dictionary=id2word_phone,
```

```
In [209]: # Show graph
          limit=12; start=3; step=1;
          x = range(start, limit, step)
          plt.plot(x, coherence_values_phone)
          plt.xlabel("Num Topics")
          plt.ylabel("Coherence score")
          plt.legend(("coherence_values"), loc='best')
          plt.show()
```




```
In [210]: # Print the coherence scores
          for m, cv in zip(x, coherence_values_phone):
              print("Num Topics =", m, " has Coherence Value of", round(cv, 4))

Num Topics = 3  has Coherence Value of 0.3517
Num Topics = 4  has Coherence Value of 0.3528
Num Topics = 5  has Coherence Value of 0.4109
Num Topics = 6  has Coherence Value of 0.4279
Num Topics = 7  has Coherence Value of 0.4214
Num Topics = 8  has Coherence Value of 0.4831
Num Topics = 9  has Coherence Value of 0.4071
Num Topics = 10  has Coherence Value of 0.4358
Num Topics = 11  has Coherence Value of 0.4317


In [211]: optimal_model_phone = model_list_phone[5]
          vis = pyLDAvis.gensim.prepare(optimal_model_phone, corpus_phone, id2word_phone)
          pyLDAvis.display(vis)

Out[211]: <IPython.core.display.HTML object>

In [212]: def format_topics_sentences(ldamodel, corpus, texts, original, vendor, sales, dutch):
              # Array of top 10 topics
              top10array = []

              for row in range(ldamodel.num_topics):
                  wp = ldamodel.show_topic(row)
                  topic_keywords = ", ".join([word for word, prop in wp])
                  top10array.append((row, topic_keywords))

              top10dict = dict(top10array)
```

```
        sent_topics_df = pd.DataFrame(pd.DataFrame([sorted(topic[0], key=lambda x: (x[1]), revers
        sent_topics_df.columns=["Data"]

        original = original.reset_index()
        sent_topics_df = pd.concat([sent_topics_df, original], axis=1)
        sent_topics_df.columns.values[2] = 'Original'

        vendors = vendor.reset_index()
        sent_topics_df = pd.concat([sent_topics_df, vendors], axis=1)
        sent_topics_df.columns.values[4] = 'Vendors'

        total_sales = sales.reset_index()
        sent_topics_df = pd.concat([sent_topics_df, total_sales], axis=1)
        sent_topics_df.columns.values[6] = 'Total_sales'

        dutch_id = dutch.reset_index()
        sent_topics_df = pd.concat([sent_topics_df, dutch_id], axis=1)
        sent_topics_df.columns.values[8] = 'Dutch_id'

        sent_topics_df['Original'] = sent_topics_df.Original
        sent_topics_df['Vendors'] = sent_topics_df.Vendors
        sent_topics_df['Total_sales'] = sent_topics_df.Total_sales
        sent_topics_df['Dutch_id'] = sent_topics_df.Dutch_id
        sent_topics_df['Index'] = sent_topics_df.index
        sent_topics_df['Dominant_Topic'] = sent_topics_df.Data.apply(lambda x: x[0])
        sent_topics_df['Perc_Contribution'] = sent_topics_df.Data.apply(lambda x: round(x[1],4))
        sent_topics_df['Topic_Keywords'] = sent_topics_df.Dominant_Topic.apply(lambda x: top10dic

        sent_topics_df = sent_topics_df[['Dominant_Topic', 'Perc_Contribution', 'Topic_Keywords',
        return(sent_topics_df)

    vendor = df2[phone]['vendor']
    sales = df2[phone]['total_sales']
    dutch = df2[phone]['dutch']
    df2_topic_sents_keywords_phone = format_topics_sentences(ldamodel=optimal_model_phone, corpus

    # Format
    df2_dominant_topic_phone = df2_topic_sents_keywords_phone.reset_index()
    df2_dominant_topic_phone.columns = ['Document_No', 'Dominant_Topic', 'Topic_Perc_Contrib', 'K

In [213]: # Group top 5 sentences under each topic
    sent_topics_sorted_phone = pd.DataFrame()

    sent_topics_outdf_grpd = df2_topic_sents_keywords_phone.groupby('Dominant_Topic')

    for i, grp in sent_topics_outdf_grpd:
        sent_topics_sorted_phone = pd.concat([sent_topics_sorted_phone,
                                              grp.sort_values(['Perc_Contribution'], ascending
                                             axis=0)

    # Reset Index
    sent_topics_sorted_phone.reset_index(drop=True, inplace=True)

    # Format
    sent_topics_sorted_phone.columns = ['Topic_Num', "Topic_Perc_Contrib", "Keywords", "Text", "I
```

```
In [215]: cluster0 = df2_dominant_topic_phone.loc[df2_dominant_topic_phone['Dominant_Topic'] == 0
          cluster1 = df2_dominant_topic_phone.loc[df2_dominant_topic_phone['Dominant_Topic'] == 1
          cluster2 = df2_dominant_topic_phone.loc[df2_dominant_topic_phone['Dominant_Topic'] == 2
          cluster3 = df2_dominant_topic_phone.loc[df2_dominant_topic_phone['Dominant_Topic'] == 3
          cluster4 = df2_dominant_topic_phone.loc[df2_dominant_topic_phone['Dominant_Topic'] == 4
          cluster5 = df2_dominant_topic_phone.loc[df2_dominant_topic_phone['Dominant_Topic'] == 5
          cluster6 = df2_dominant_topic_phone.loc[df2_dominant_topic_phone['Dominant_Topic'] == 6
          cluster7 = df2_dominant_topic_phone.loc[df2_dominant_topic_phone['Dominant_Topic'] == 7

          d = {'Cluster_No' : pd.Series([0,1,2,3,4,5,6,7]),
               'Total_listings' : pd.Series([len(df2_dominant_topic_phone)]),
               'Numb_of_list ' : pd.Series([len(cluster0), len(cluster1), len(cluster2), len(clus
               'Total_uniq_vendors' : pd.Series([df2_dominant_topic_phone.Vendors.nunique()]),
               'Unique_vendors' : pd.Series([cluster0.Vendors.nunique(),cluster1.Vendors.nunique(
               'Total_revenue' : pd.Series([df2_dominant_topic_phone.Total_sales.sum()]),
               'Revenue' : pd.Series([cluster0.Total_sales.sum(),cluster1.Total_sales.sum(),clust
              'Dutch_identifier' : pd.Series([cluster0.Dutch_id.sum(),cluster1.Dutch_id.sum(),clu
          df_intel = pd.DataFrame(d)
          cols = df_intel.columns.tolist()
          cols = ['Cluster_No',
           'Total_listings',
           'Numb_of_list ',
           'Total_uniq_vendors',
           'Unique_vendors',
           'Total_revenue',
           'Revenue',
           'Dutch_identifier']
          df_intel = df_intel[cols]
```

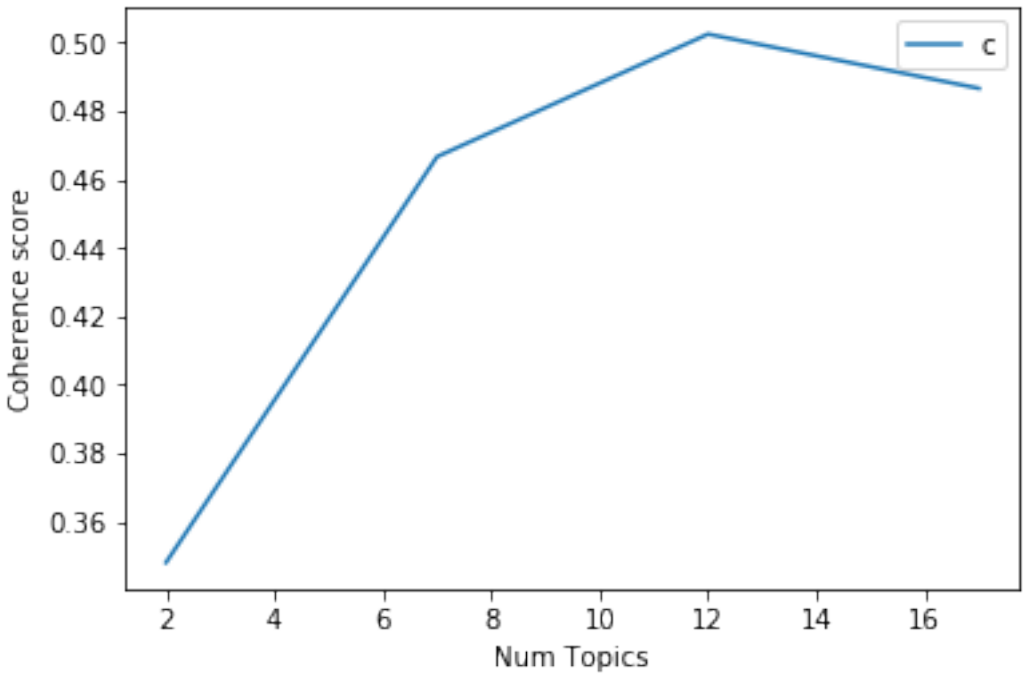
```
In [216]: writer = ExcelWriter('D:/fegmiedema/Desktop/phone.xlsx')
          sent_topics_sorted_phone.to_excel(writer,'Sheet1',index=False)
          df_intel.to_excel(writer,'Sheet2',index=False)
          writer.save()
```

B.2.17. Website - 670 listings - 12 clusters

```
In [217]: data_lemmatized_website = lemmatization(df2[website]['title&description'])
          id2word_website = corpora.Dictionary(data_lemmatized_website)
          texts_website = data_lemmatized_website
          corpus_website = [id2word_website.doc2bow(text) for text in texts_website]
```

```
In [220]: model_list_website, coherence_values_website= compute_coherence_values(dictionary=id2wo
```

```
In [221]: # Show graph
          limit=22; start=2; step=5;
          x = range(start, limit, step)
          plt.plot(x, coherence_values_website)
          plt.xlabel("Num Topics")
          plt.ylabel("Coherence score")
          plt.legend(("coherence_values"), loc='best')
          plt.show()
```




```
In [223]: # Print the coherence scores
          for m, cv in zip(x, coherence_values_website):
              print("Num Topics =", m, " has Coherence Value of", round(cv, 4))

Num Topics = 2  has Coherence Value of 0.3482
Num Topics = 7  has Coherence Value of 0.4666
Num Topics = 12  has Coherence Value of 0.5023
Num Topics = 17  has Coherence Value of 0.4865
```

```
In [224]: optimal_model_website = model_list_website[2]
          vis = pyLDAvis.gensim.prepare(optimal_model_website, corpus_website, id2word_website)
          pyLDAvis.display(vis)

Out[224]: <IPython.core.display.HTML object>
```

```
In [225]: def format_topics_sentences(ldamodel, corpus, texts, original, vendor, sales, dutch):
              # Array of top 10 topics
              top10array = []

              for row in range(ldamodel.num_topics):
                  wp = ldamodel.show_topic(row)
                  topic_keywords = ", ".join([word for word, prop in wp])
                  top10array.append((row, topic_keywords))

              top10dict = dict(top10array)

              sent_topics_df = pd.DataFrame(pd.DataFrame([sorted(topic[0], key=lambda x: (x[1]), revers
              sent_topics_df.columns=["Data"]

              original = original.reset_index()
```

```
        sent_topics_df = pd.concat([sent_topics_df, original], axis=1)
        sent_topics_df.columns.values[2] = 'Original'

        vendors = vendor.reset_index()
        sent_topics_df = pd.concat([sent_topics_df, vendors], axis=1)
        sent_topics_df.columns.values[4] = 'Vendors'

        total_sales = sales.reset_index()
        sent_topics_df = pd.concat([sent_topics_df, total_sales], axis=1)
        sent_topics_df.columns.values[6] = 'Total_sales'

        dutch_id = dutch.reset_index()
        sent_topics_df = pd.concat([sent_topics_df, dutch_id], axis=1)
        sent_topics_df.columns.values[8] = 'Dutch_id'

        sent_topics_df['Original'] = sent_topics_df.Original
        sent_topics_df['Vendors'] = sent_topics_df.Vendors
        sent_topics_df['Total_sales'] = sent_topics_df.Total_sales
        sent_topics_df['Dutch_id'] = sent_topics_df.Dutch_id
        sent_topics_df['Index'] = sent_topics_df.index
        sent_topics_df['Dominant_Topic'] = sent_topics_df.Data.apply(lambda x: x[0])
        sent_topics_df['Perc_Contribution'] = sent_topics_df.Data.apply(lambda x: round(x[1
        sent_topics_df['Topic_Keywords'] = sent_topics_df.Dominant_Topic.apply(lambda x: to

        sent_topics_df = sent_topics_df[['Dominant_Topic', 'Perc_Contribution', 'Topic_Keyw
        return(sent_topics_df)

    vendor = df2[website]['vendor']
    sales = df2[website]['total_sales']
    dutch = df2[website]['dutch']
    df2_topic_sents_keywords_website = format_topics_sentences(ldamodel=optimal_model_websi

    # Format
    df2_dominant_topic_website = df2_topic_sents_keywords_website.reset_index()
    df2_dominant_topic_website.columns = ['Document_No', 'Dominant_Topic', 'Topic_Perc_Cont
```

In [226]:
```
    # Group top 5 sentences under each topic
    sent_topics_sorted_website = pd.DataFrame()

    sent_topics_outdf_grpd = df2_topic_sents_keywords_website.groupby('Dominant_Topic')

    for i, grp in sent_topics_outdf_grpd:
        sent_topics_sorted_website = pd.concat([sent_topics_sorted_website,
                                                grp.sort_values(['Perc_Contribution'], asc
                                               axis=0)

    # Reset Index
    sent_topics_sorted_website.reset_index(drop=True, inplace=True)

    # Format
    sent_topics_sorted_website.columns = ['Topic_Num', "Topic_Perc_Contrib", "Keywords", "T
```

In [227]:
```
    cluster0 = df2_dominant_topic_website.loc[df2_dominant_topic_website['Dominant_Topic']
    cluster1 = df2_dominant_topic_website.loc[df2_dominant_topic_website['Dominant_Topic']
    cluster2 = df2_dominant_topic_website.loc[df2_dominant_topic_website['Dominant_Topic']
    cluster3 = df2_dominant_topic_website.loc[df2_dominant_topic_website['Dominant_Topic']
    cluster4 = df2_dominant_topic_website.loc[df2_dominant_topic_website['Dominant_Topic']
```

```
cluster5 = df2_dominant_topic_website.loc[df2_dominant_topic_website['Dominant_Topic'] == 5]
cluster6 = df2_dominant_topic_website.loc[df2_dominant_topic_website['Dominant_Topic'] == 6]
cluster7 = df2_dominant_topic_website.loc[df2_dominant_topic_website['Dominant_Topic'] == 7]
cluster8 = df2_dominant_topic_website.loc[df2_dominant_topic_website['Dominant_Topic'] == 8]
cluster9 = df2_dominant_topic_website.loc[df2_dominant_topic_website['Dominant_Topic'] == 9]
cluster10 = df2_dominant_topic_website.loc[df2_dominant_topic_website['Dominant_Topic'] == 10
cluster11 = df2_dominant_topic_website.loc[df2_dominant_topic_website['Dominant_Topic'] == 11

d = {'Cluster_No' : pd.Series([0,1,2,3,4,5,6,7,8,9,10,11]),
     'Total_listings' : pd.Series([len(df2_dominant_topic_website)]),
     'Numb_of_list ' : pd.Series([len(cluster0), len(cluster1), len(cluster2), len(cluster3),
     'Total_uniq_vendors' : pd.Series([df2_dominant_topic_website.Vendors.nunique()]),
     'Unique_vendors' : pd.Series([cluster0.Vendors.nunique(),cluster1.Vendors.nunique(),clus
     'Total_revenue' : pd.Series([df2_dominant_topic_website.Total_sales.sum()]),
     'Revenue' : pd.Series([cluster0.Total_sales.sum(),cluster1.Total_sales.sum(),cluster2.To
    'Dutch_identifier' : pd.Series([cluster0.Dutch_id.sum(),cluster1.Dutch_id.sum(),cluster2.
df_intel = pd.DataFrame(d)
cols = df_intel.columns.tolist()
cols = ['Cluster_No',
 'Total_listings',
 'Numb_of_list ',
 'Total_uniq_vendors',
 'Unique_vendors',
 'Total_revenue',
 'Revenue',
 'Dutch_identifier']
df_intel = df_intel[cols]
```

In [228]:
```
writer = ExcelWriter('D:/fegmiedema/Desktop/website.xlsx')
sent_topics_sorted_website.to_excel(writer,'Sheet1',index=False)
df_intel.to_excel(writer,'Sheet2',index=False)
writer.save()
```

Bibliografie

[1] Andres Baravalle, Mauro Sanchez Lopez, and Sin Wee Lee. Mining the Dark Web. *IEEE Conference Publication*, pages 350–356, 2016. doi: 10.1109/ICDMW.2016.149.

[2] Belastingdienst. Handhavingsarrangement 2016-2019, May 2016. URL InterndocumentBelastingdienst.

[3] Peter Biddle, Paul England, Marcus Peinado, and Bryan Willman. The darknet and the future of content protection. *Digital Rights Management, Lecture Notes in Computer Science*, 2696:155–176, 11 2002.

[4] David M Blei, Blei@cs Berkeley Edu, Andrew Y Ng, Ang@cs Stanford Edu, Michael I Jordan, and Jordan@cs Berkeley Edu. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3: 993–1022, 2003. ISSN 15324435. doi: 10.1162/jmlr.2003.3.4-5.993.

[5] J. Broséus, D. Rhumorbarbe, C. Mireault, V. Ouellette, F. Crispino, and D. Décary-Hétu. Studying illicit drug trafficking on Darknet markets: Structure and organisation from a Canadian perspective. *Forensic Science International*, 264:7–14, 2016. ISSN 18726283. doi: 10.1016/j.forsciint.2016.02.045.

[6] CBS. Cbs statline, Jul 2018. URL http://statline.cbs.nl/.

[7] Zhiyuan Cheng, James Caverlee, and Kyumin Lee. You Are Where You Tweet : A Content-Based Approach to Geo-locating Twitter Users. *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, pages 759–768, 2010. ISSN 15461718. doi: 10.1145/1871437.1871535.

[8] N. Christin. An EU-focused analysis of drug supply on the online anonymous marketplace ecosystem. *European Monitoring Centre for Drugs and Drug Addiction (EMCDDA)*, pages 1–34, 2017. URL http://www.emcdda.europa.eu/system/files/attachments/6624/EU-focused-analysis-of-drug-supply-on-the-anomymous-online-marketplace.pdf.

[9] Nicolas Christin. Traveling the silk road: a measurement analysis of a large anonymous online marketplace. *WWW 2013*, pages 213–224, 05 2013.

[10] Vincenzo Ciancaglini, Marco Balduzzi, Robert Mcardle, and Martin Rösler. Exploring the Deep Web Contents. *Trend Micro*, pages 5–6, 2013. URL https://www.trendmicro.com/cloud-content/us/pdfs/security-intelligence/white-papers/wp{_}below{_}the{_}surface.pdf.

[11] Thomas Colin. Extracting topics from 11000 newsgroups posts with python gensim and lda, Jun 2018. URL https://dataskunkworks.com/2018/06/06/extracting-topics-from-11000-newsgroups-posts-with-python-gensim-and-lda/.

[12] Joseph Cox. 7 ways the cops will bust you on the dark web, Jun 2016. URL https://motherboard.vice.com/en_us/article/vv73pj/7-ways-the-cops-will-bust-you-on-the-dark-web.

[13] Janis Dalins, Campbell Wilson, and Mark Carman. Criminal motivation on the dark web: A categorisation model for law enforcement. *Digital Investigation*, 01 2018.

[14] Martin Dittus, Joss Wright, and Mark Graham. Platform Criminalism: The 'Last-Mile' Geography of the Darknet Market Supply Chain. *IW3C2*, 2017. doi: 10.1145/3178876.3186094. URL http://arxiv.org/abs/1712.10068{%}0Ahttp://dx.doi.org/10.1145/3178876.3186094.

[15] Diana S. Dolliver. Evaluating drug trafficking on the Tor Network: Silk Road 2, the sequel. *International Journal of Drug Policy*, 26(11):1113–1123, 2015. ISSN 18734758. doi: 10.1016/j.drugpo.2015.01.008. URL http://dx.doi.org/10.1016/j.drugpo.2015.01.008.

[16] A. Feelders, H. Daniels, and M. Holsheimer. Methodological and practical aspects of data mining. *Information and Management*, 37(5):271–281, 2000. ISSN 03787206. doi: 10.1016/S0378-7206(99)00051-8.

[17] Ronan Feldman and James Sanger. *The text mining Handbook: advanced approaches in analyzing unstructured data*. University Press, 2007.

[18] FIOD. Jaarcontract 2017, 2017. URL InterndocumentFIOD.

[19] FIOD. Jaarcontract 2018, 2017. URL InterndocumentFIOD.

[20] European Monitoring Centre for Drugs, Drug Addiction, and Europol. Drugs and the darknet: Perspectives for enforcement, research and policy. Technical report, European Monitoring Centre for Drugs and Drug Addiction and Europol, 2017. URL http://www.emcdda.europa.eu/system/files/publications/6585/TD0417834ENN.pdf.

[21] Gwern. Darknet market archives (2013-2015), Dec 2013. URL https://www.gwern.net/DNM-archives.

[22] Bo Han, Paul Cook, and Timothy Baldwin. Text-based twitter user geolocation prediction. *Journal of Artificial Intelligence Research*, 49:451–500, 2014. ISSN 10769757. doi: 10.1613/jair.4200.

[23] Kernstof. productgroep - de betekenis volgens marketing kernstof, May 2017. URL https://www.ensie.nl/marketing-kernstof/productgroep.

[24] N. Kop and P. Klerks. *Intelligencegestuurd politiewerk*. Politieacademie, 2009.

[25] Susan Li. Topic modeling in python with nltk and gensim, Apr 2018. URL https://datascienceplus.com/topic-modeling-in-python-with-nltk-and-gensim/.

[26] Maarten Lörtzer. Illegale handel en wandel op darkweb flink geraakt door recente politieactie, Aug 2017. URL https://www.tno.nl/nl/over-tno/nieuws/2017/8/illegale-handel-en-wandel-op-darkweb-flink-geraakt-door-recente-politieactie/.

[27] Eric Malmi, Arno Solin, and Aristides Gionis. The blind leading the blind: Network-based location estimation under uncertainty. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9285:406–421, 2015. ISSN 16113349. doi: 10.1007/978-3-319-23525-7_25.

[28] James Martin. Lost on the Silk Road: Online drug distribution and the 'cryptomarket'. *Criminology & Criminal Justice*, 14(3):351–367, 2014. doi: 10.1177/1748895813505234.

[29] David Mimno, Hanna M. Wallach, Edmund Talley, Miriam Leenders, and Andrew McCallum. Optimizing semantic coherence in topic models. *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 262–272, 2011. ISSN 1937284115.

[30] Daniel Moore and Thomas Rid. Cryptopolitik and the Darknet. *Survival. Global Politics and Strategy*, 58(1):7–38, 2016. ISSN 0039-6338\r1468-2699. doi: 10.1080/00396338.2016.1142085. URL http://www.tandfonline.com/doi/full/10.1080/00396338.2016.1142085.

[31] Rasmus Munksgaard, Jakob Demant, and Gwern Branwen. A replication and methodological critique of the study "evaluating drug trafficking on the tor network". *International Journal of Drug Policy*, 03 2016.

[32] Politieacademie. verticale fraude, 2018. URL https://thesaurus.politieacademie.nl/Thesaurus/Term/7452.

[33] Tor Project. Welcome to tor metrics!, 2018. URL https://metrics.torproject.org/.

[34] Damien Rhumorbarbe, Ludovic Staehli, Julian Broséus, Quentin Rossy, and Pierre Esseiva. Buying drugs on a Darknet market: A better deal? Studying the online illicit drug market through the analysis of digital, physical and chemical data. *Forensic Science International*, 267:173–182, 2016. ISSN 18726283. doi: 10.1016/j.forsciint.2016.08.032.

[35] Digital Shadows. The state of cybercrime in the post-alphabay and hansa age, 2018. URL https://resources.digitalshadows.com/whitepapers-and-reports/the-state-of-cybercrime-in-the-post-alphabay-and-hansa-age.

[36] Carson Sievert and Kenneth Shirley. LDAvis: A method for visualizing and interpreting topics. *Proceedings of the Workshop on Interactive Language Learning, Visualization, and Interfaces*, pages 63–70, 2014. ISSN 10495258. doi: 10.1.1.100.1089. URL http://www.aclweb.org/anthology/W/W14/W14-3110.

[37] Kyle Soska and Nicolas Christin. Measuring the Longitudinal Evolution of the Online Anonymous Marketplace Ecosystem. *24th USENIX Security Symposium*, pages 33–48, 2015. ISSN 978-1-931971-232. URL https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/soska.

[38] Mike Thelwall, Liwen Vaughan, and Lennart Bjorneborn. Webometrics. *Annual Reveiw of Information Science and Technology*, pages 81–135, 2005. ISSN 00664200. doi: 10.1002/aris.1440390110.

[39] R S van Wegberg, Samaneh Tajalizadehkhoob, Kyle Soska, Ugur Akyazi, Carlos Ganan, A J Klievink, Nicolas Christin, and Michel J G Van Eeten. Plug and Prey ? Measuring the Commoditization of Cybercrime via Online Anonymous Markets. Unpublished, 2018.

[40] Jeff Wallenfeldt. Favela, Jul 2016. URL https://www.britannica.com/topic/favela.

[41] ChengXiang Zhai and Sean Massung. *Text data management and analysis: a practical introduction to information retrieval and text mining*. ACM Books, 2016.