Table of appendix

Appendix 1. List of extra stop words

	Words						
1	Kan	12	Gøre	23	Bruge	34	Fordi
2	Så	13	Går	24	Dag	35	Gå
3	Få	14	Bla.	25	Sige	36	Bare
4	Se	15	Mest	26	Vores	37	Lidt
5	Ved	16	Gør	27	Komme	38	Sætte
6	Ser	17	Stor	28	Siger	39	Of
7	Hvordan	18	Del	29	Sagde	40	On
8	Mere	19	Nå	30	Ny	41	The
9	Nye	20	Både	31	Mellem		
10	Derfor	21	Tæt	32	Omkring		
11	Får	22	Andre	33	Pga.		

Appendix 2. Immersion Journal

Appendix 3. Example on collaboration and not collaboration

Example of "not a collaboration"

Example of "collaboration"

Appendix 4. VNA Network measures

Top 10	*In-degree centrality score*	*Out-degree centrality score*	*Closeness centrality score*	*Betweeness centrality score*
Red Barnet	14	15	0.69	0.16
Amnesty	12	4	0.61	0.14
Danmarks Indsamling	12	0	0.57	0.09
Røde Kors	9	13	0.65	0.09
Mellemfolkeligt Samvirke	9	5	0.57	0.13
Plan Børnefonden	8	4	0.56	0.10
Oxfam IBIS	8	10	0.60	0.06
Dansk Flygtningehjælp	7	9	0.57	0.03
Globalt Fokus	7	5	0.55	0.06
DIGNITY	6	5	0.56	0.01
Folkekirkens Nødhjælp	6	13	0.61	0.15
Care Dnmark	5	3	0.53	0.01
Kirkens Korshær	3	0	0.44	0.01
Refugees Welcome Danmark	3	7	0.49	0.00
Læger Uden Grænser	3	6	0.49	0.01
WFP Danmark	3	0	0.44	0.00
Global Action	3	0	0.41	0.00
Danmission	3	1	0.40	0.00
Caritas	2	3	0.44	0.01
Repatriate the Children Denmark	2	3	0.47	0.00
Børns Vilkår	2	2	0.44	0.00
UNICEF Danmark	2	5	0.48	0.01
SOS Børnebyerne	2	1	0.44	0.00
92-gruppen	1	0	0.38	0.00
Mission Øst	1	1	0.40	0.00
Menneskeret	1	3	0.42	0.01
Adra Danmark	1	9	0.57	0.07
LGBT Asylum	1	3	0.45	0.00
Dansk Folkehjælp	0	2	0.41	0.00
Sæt Dem Fri	0	1	0.38	0.00
UNDP Danmark	0	1	0.36	0.00
Blå Kors	0	2	0.39	0.00

Appendix 5. Retweet SNA network measures

Top 10	*In-degree centrality score*	*Out-degree centrality score*	*Closeness centrality score*	*Betweeness centrality score*
RasmusPrehn	13	0	0.43	0.06
DanishMFA	10	0	0.40	0.06
Globaltfokus	9	44	0.43	0.12
Udviklingsmin	8	0	0.37	0.02
redbarnetdk	7	66	0.45	0.21
Dr2deadline	7	0	0.37	0.01
AndersLadekarl	7	0	0.35	0.01
DRC_dk	6	42	0.41	0.09
Denmark_UN	6	0	0.36	0.01
politiken	6	0	0.38	0.01

Appendix 6. Small retweet SNA network measures

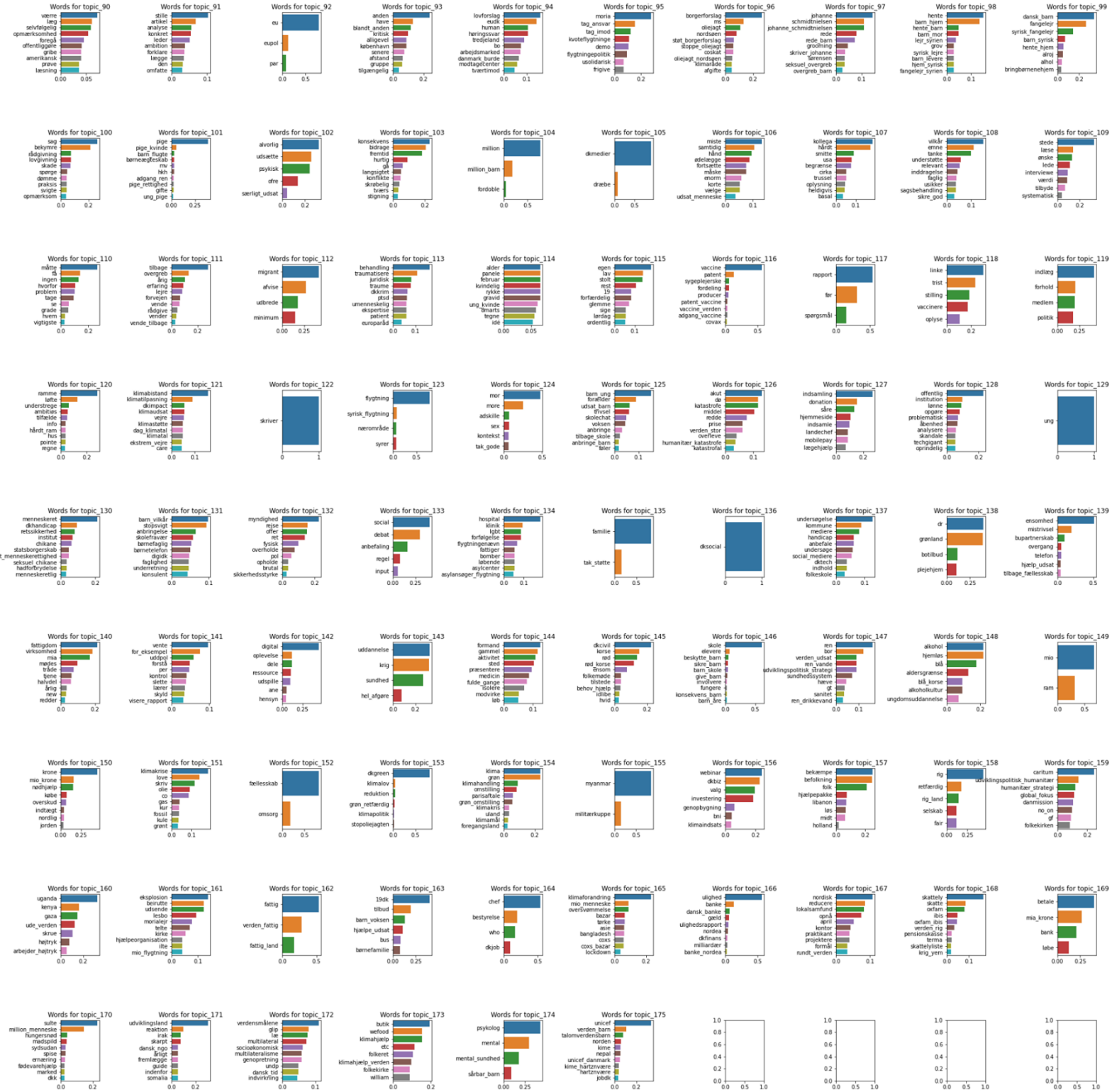
Top 10	*In-degree centrality score*	*Out-degree centrality score*	*Closeness centrality score*	*Betweeness centrality score*
Globaltfokus	9	6	0.55	0.15
Redbarnetdk	7	7	0.64	0.35
Ofamibis	6	5	0.53	0.08
DRC_dk	6	6	0.53	0.16
Menneskeret	4	1	0.34	0.01
DignityDK	4	5	0.48	0.03
UNDP_Danmark	3	2	0.39	0.01
Amnestydk	3	4	0.48	0.03
WFP_DK	3	2	0.37	0.00
CARE_Danmark	3	3	0.45	0.00
RefWelcome	2	2	0.38	0.01
UNICEFDK	2	1	0.38	0.00
ActionAidDK	2	7	0.48	0.10
PlanBornefonden	2	4	0.48	0.04
Danskrodekors	2	0	0.42	0.00
Lgbt_asylm	2	3	0.36	0.00
Msf_dk	2	0	0.36	0.00
Noedhjaelp	2	2	0.35	0.00
BornsVilkår	2	1	0.42	0.08
AVestegnen	2	4	0.44	0.03
Danmissiondk	2	0	0.04	0.00
RTC_DK	1	2	0.39	0.00
Blaakorsdanmark	0	1	0.29	0.00
CaritasDanmark	0	1	0.04	0.00

Appendix 7. LDA-parameters for each topic

	actor	tweet	date	proc_text_all	hashtags_clean	topic_0	topic_1	topic_2	topic_3	topic_4
0	PlanBornefonden	13-årige Larissa bor i Sahel-regionen, og var ...	[årig_larissa, larissa_bor, bor_sahelregion, s...	årig larissa bor sahelregion tvinge flygte lan...	0.000000	0.000000	0.992848	0.0	0.000000
1	PlanBornefonden	Vi ønsker alle muslimer en god Eid i aften! Ei...	[ønske_muslim, muslim_god, god_eid, eid_aften,...	ønske muslim god eid aften eid mubarak ønske_m...	0.000000	0.000000	0.000000	0.0	0.976431
2	PlanBornefonden	Kom til samtalekøkken med @BosseStine og @Clau...	[komme_samtal ekøkken, samtalekøkken_ maj, maj_s...	komme samtalekøkken maj spise lækker ret menu ...	0.000000	0.000000	0.000000	0.0	0.989029
3	PlanBornefonden	Mali, Burkina Faso og Niger - også kendt som d...	[mali_burkina, burkina_faso, faso_niger, niger...	mali burkina faso niger kendt central sahele s...	0.000000	0.994836	0.000000	0.0	0.000000
4	PlanBornefonden	Vores seje kollega Iben Østergaard Markussen f...	[sej_kollega, kollega_ibe, ibe_østergaard, øst...	sej kollega ibe østergaard markusse fortæller ...	0.993019	0.000000	0.000000	0.0	0.000000

For further info, look at appendix 17.

Appendix 8. hSBM 121 topics (continue next page)


Words for topic_90
værre læg selvfølgelig opmærksomhed foregå offentliggøre gribe amerikansk prøve læsning
Words for topic_91
stille artikel analyse konkret leder ambition forklare lægge den omfatte
Words for topic_92
eu eupol par
Words for topic_93
anden have blandt_andre kritisk alligevel københavn senere afstand gruppe tilgængelig
Words for topic_94
lovforslag eudk human høringssvar tredjeland bo arbejdsmarked danmark_burde modtageCenter tværtimod
Words for topic_95
moria tag_ansvar tag_imod kvoteflygtninge demo flygtningepolitik usolidarisk frigive
Words for topic_96
borgerforslag ms oliejagt nordsøen stat_borgerforslag stoppe_oliejagt coskat oliejagt_nordsøen klimaråde afgifte
Words for topic_97
johanne schmidtnielsen johanne_schmidtnielsen rede groolming skriver_johanne sørensen seksuel_overgreb overgreb_barn
Words for topic_98
hente barn_hjem hente_barn barn_mor lejr_syrien grov syrisk_lejre barn_ilvere hjem_syrisk fangelejr_syrien
Words for topic_99
dansk_barn fangelejr syrisk_fangelejr barn_syrisk hente_hjem afrej alhol bringbørnenehjem
Words for topic_100
sag bekymre rådgivning lovgivning skade spørge damme praksis svigte opmærksom
Words for topic_101
pige pige_kvinde barn_flugte børneægteskab mv hih adgang_ren pige_rettighed gifte ung_pige
Words for topic_102
alvorlig udsætte psykisk ofre særligt_udsat
Words for topic_103
konsekvens bidrage fremtid hurtig gå langsigtet konflikte skrøbelig tværs stigning
Words for topic_104
million million_barn fordoble
Words for topic_105
dkmedier dræbe
Words for topic_106
miste samtidig hånd adelægge fortsætte måske enorm korte vælge udsat_mennesker
Words for topic_107
kollega hårdt smitte usa begrænse cirka trussel oplysning heldigvis basal
Words for topic_108
vilkår emne tanke understøtte relevant inddragelse faglig usikker sagsbehandling sikre_god
Words for topic_109
stede læse ønske lede interviewe værdi tilbyde systematisk
Words for topic_110
måtte få ingen hvorfor problem tage se grade hvem vigtigste
Words for topic_111
tilbage overgreb årig erfaring lejre forvejen vende rådgive vender vende_tilbage
Words for topic_112
migrant afvise udbrede minimum
Words for topic_113
behandling traumatisere juridisk traume dkkrim ptsd umenneskelig ekspertise patient europaråd
Words for topic_114
alder panele februar kvindelig rykke gravid ung_kvinde tilmarts tegne idé
Words for topic_115
egen lav stolt rest 19 forfærdelig glemme sige ordentlig
Words for topic_116
vaccine patent sygeplejerske fordeling producer patent_vaccine vaccine_verden adgang_vaccine covax
Words for topic_117
rapport før spørgsmål
Words for topic_118
linke trist stilling vaccinere oplyse
Words for topic_119
indlæg forhold medlem politik
Words for topic_120
ramme løfte understrege ambitiøs tilfælde info hårdt_ram hus pointe regne
Words for topic_121
klimabistand klimatilpasning dkimpact klimaudsat vejre klimastøtte dag_klimatal klimatal ekstrem_vejre care
Words for topic_122
skriver
Words for topic_123
flygtning syrisk_flygtning nærområde syrer
Words for topic_124
mor more adskille sex kontekst tak_gode
Words for topic_125
barn_ung forælder udsat_barn tfivsel skolechat voksen anbringe tilbage_skole anbringe_barn fælles
Words for topic_126
akut da katastrofe middel redde prise verden_stor overleve humanitær_katastrofe katastrofal
Words for topic_127
indsamling donation såre hjemmeside indsamle landechef mobilepay lægehjælp
Words for topic_128
offentlig institution lænke opgøre problematisk åbenhed analysere skandale techgigant oprindelig
Words for topic_129
ung
Words for topic_130
menneskeret dkhandicap retssikkerhed institut chikane statsborgerskab ret_menneskerettighed seksuel_chikane hadforbrydelse menneskeretlig
Words for topic_131
barn_vilkår støjsvigt anbringelse skolefravær børnefaglig børnetelefon digidk faglighed underretning konsulent
Words for topic_132
myndighed mise offer ret fysisk overholde pol opholde brutal sikkerhedsstyrke
Words for topic_133
social debat anbefaling regel input
Words for topic_134
hospital klinik lgbt forfølgelse flygtningenævn fattiger bomber løbende asylcenter asylansøger_flygtning
Words for topic_135
familie tak_støtte
Words for topic_136
dksocial
Words for topic_137
undersøgelse kommune mediere handicap anbefale undersøge social_mediere diktech indhold folkeskole
Words for topic_138
dr grønland botilbud plejehjem
Words for topic_139
ensomhed mistrivsel bupartnerskab overgang telefon hjælp_udsat tilbage_fællesskab
Words for topic_140
fattigdom virksomhed mia mødes tråde tjene halvdel årlig new redder
Words for topic_141
vente for_eksempel usdpol forstå per kontrol slette lærer skyld visere_rapport
Words for topic_142
digital opleve dele ressource udspille ane hensyn
Words for topic_143
uddannelse krig sundhed hel_afgøre
Words for topic_144
formand gammel aktivitet sted præsentere medicin fulde_gange isolere modvirke løb
Words for topic_145
dkcivil korse rød rød_korse ensom folkemøde tilstede behov_hjælp idébe hvid
Words for topic_146
skole elevere beskytte_barn sikre_barn barn_skole give_barn involvere fungere konsekvens_barn barn_øre
Words for topic_147
ren bor verden_udsat ren_vande udviklingspolitisk_strategi sundhedssystem hæve gf sanitet ren_drikkevand
Words for topic_148
alkohol hjemløs blå aldersgrænse blå_korse alkoholkultur ungdomsuddannelse
Words for topic_149
mio ram
Words for topic_150
krone mio_krone nedhjælp købe overskud indtægt nordlig jorden
Words for topic_151
klimakrise love skrive olie co gas kur fossil kulle grønt
Words for topic_152
fællesskab omsorg
Words for topic_153
dkgreen klimalov reduktion grøn_rettfærdig klimapolitik stopoliejagten
Words for topic_154
klima grøn klimahandling omstilling parisaftale grøn_omstilling klimakris uland klimamål foregangsland
Words for topic_155
myanmar militærkuppe
Words for topic_156
webinar dkbiz valg investering genopbygning bni klimaindsats
Words for topic_157
bekæmpe befolkning folk hjælpepakke libanon løs midt holland
Words for topic_158
rig retfærdig rig_land seiskab fair
Words for topic_159
udviklingspolitisk_humanitær caritum humanitær_strategi global_fokus danmission no_on gf folkekirken
Words for topic_160
uganda kenya gaza ude_verden skrue højtryk arbejder_højtryk
Words for topic_161
eksplosion beirutby udsende lesbo morallejr telte kirke hjælpeorganisation ilte mio_flygtning
Words for topic_162
fattig verden_fattig fattig_land
Words for topic_163
19dk tilbud barn_voksen hjælpe_udsat bus børnefamilie
Words for topic_164
chef bestyrelse who dkjob
Words for topic_165
klimaforandring mio_menneske oversvømmelse bazar tørke asie bangladesh coxs coxs_bazar lockdown
Words for topic_166
ulighed banke dansk_banke gæld ulighedsrapport nordea dkfinans milliardær banke_nordea
Words for topic_167
nordisk eduscere lokalsamfund opnå april kontor praktikant projektere formål rundt_verden
Words for topic_168
skattely skatte oxfam ibis oxfam_ibis verden_rig pensionskasse terma skattelyliste krig_yem
Words for topic_169
betale mia_krone bank løbe
Words for topic_170
sulte million_menneske hungersnød madspild sydsudan spise ernæring fødevarehjælp marked dkk
Words for topic_171
udviklingsland reaktion irak skærpe dansk_ngo årligt fremlægge guide indenfor somalia
Words for topic_172
verdensmålene glip læ multilateral socioøkonomisk multilateralisme genopretning undp dansk_tid indvirkning
Words for topic_173
butik nerfood klimahjælp etc folkeret klimahjælp_verden folkekirke william
Words for topic_174
psykolog mental mental_sundhed sårbar_barn
Words for topic_175
unicef verden_barn talomverdensbarn norden kime nepal unicef_danmark kime_hårtznvære hårtznvære jobdk

Appendix 9. Part of Speech, 20 most used words (of nouns, verbs, and adjectives)

Top 10	*Nouns*		*Verbs*		*Adj*	
1	Barn	2630	Bruge	689	Ung	676
2	Verden	1159	Støtte	523	Vigtig	648
3	Land	821	Komme	445	God	624
4	Tak	803	Udsætte	444	Dansk	614
5	Åre	658	Give	442	Stor	601
6	Danmark	619	Skrive	418	Flere	523
7	Arbejde	448	Gøre	373	Hele	504
8	Flygtning	429	Læsse	369	Menneske	420
9	Hjælp	417	Stå	364	Fattig	335
10	Kvinde	393	Hjælpe	247	Global	291
11	Regering	375	Arbejde	246	Gode	287
12	Familie	350	Fortælle	223	Sikre	287
13	Fokus	346	Afgøre	219	Kæmpe	278
14	Behov	345	Sætte	205	Internatinoal	271
15	Skole	340	Sende	203	Lille	270
16	Tid	328	Holde	200	Hjælpe	217
17	Rapprot	318	Sulte	180	Klare	215
18	Kris	312	Finde	172	Stærk	209
19	Dag	308	Øge	157	Svær	190
20	Samarbejde	307	Handle	156	Pige	186

Appendix 10. Optimal hyperparameter for classifier

```
#Finding the best performing model and saving it
bestlasso = lasso_result.best_estimator_

print('Here we se the parameters of the best Lasso model:',lasso_result.best_params_,
      '\nAvg. accuracy score of the best performing model:',lasso_result.best_score_)
```

```
Here we se the parameters of the best Lasso model: {'lasso__C': 5, 'tfidf__use_idf': False}
Avg. accuracy score of the best performing model: 0.9581516039849373
```

For further information look appendix 20.

Appendix 11. Code book for sub-set Twitter data set

This code book is made of a subset of our entire twitter dataset and has been qualitatively and manually hand coded with an inductive and open coding approach. The subset consists of 261 tweets, consisting of about 10 tweets per organization, though some organizations did not tweet 10 times in our time period, so they are represented by less tweets in this dataset and the original. The codes that are in bold are the top codes (also referred to as nodes in Nvivo 12) and the top 10 most referenced top codes will be used to code the tweets one more time using a closed coding approach. The codes are: Børn, Menneskeretigheder, Corona, Nødhjælp, Lande, Flygtninge, Klima, Udsatte, Kritik, politk (in this order, most refferenced first).

Name	Description	Files	References
Børn (Children)	Every time an NGO tweets about a child, multiple children, a specific group of children or children in general. It can contain topics such as below though this is not an exhaustive list:	1	53

Adgang til undervisning	Access to education	1	2
Børn i Danmark	Children in Denmark	1	5
Børn på flugt	Children on the run	1	4
Børne grooming	Child grooming	1	1
Børne handleplaner	Plan of action for children	1	1
Børne hjælp	Children help	1	8
Børne sult	Starvation amongst children	1	2
Børnefællesskab	Child community	1	1
Børnesoldater	Child soldiers	1	1
Danske børn i Syrien	Danish children in Syria	1	9
Ensomhed	Lonelyness	1	1
Juridisk kønsskifte	Legal sex change	1	1
Skole	School	1	1
Social kontrol	Social control	1	1
Sommerlejr	Summer camp	1	3
Svigt	Betrayal	1	1
Syriske børn	Syrian Children	1	1
Syriske flygtningebørn	Syrian refugee children	1	3
Tilbage i skole	Back to school	1	2
Traumatiserede børn	Traumatized children	1	1
Uddannelse	Education	1	1
Udsatte børn	Vulnerable children	1	3
Corona	Everything related to corona (and when we say corona, we mean the world vide pandemic/crisis that is related to the outbreak of coronavirus, not included in this topic would be the Corona beer).	1	35
Corona blokerer	Corona is blocking	1	1
Corona støtte	Corona support	1	2
Corona tal	Corona numbers	1	1
Corona-karantæne	Corona quarantine	1	1
Coronahotline	Corona hotline	1	1
Coronakrise	Corona crisis	1	5
Coronapandemi	Corona pandemic	1	2

Coronavirus	Coronavirus	1	10
Fejle	To fail	1	1
Globalkrise	Global crisis	1	1
Modstandskraft	Resilience	1	1
Socio-økonomiske konsekvenser	Socio-economic consequences	1	1
Tuberkulose	Tuberculosis	1	1
Vaccine	Vaccine	1	5
Drop Patenter	Drop the patents	1	2
Fordele vacciner	Distribution of vaccines	1	1
Vaccinesolidaritet	Vaccine solidarity	1	1
Økonomiske konsekvenser	Economic consequences	1	1
Digitalt	Everything that pertains to digital behaviour, the digital world or digital future. This topic includes everything from digital harassment to digital finance, but it is important for this topic that it is about the digital world and not something digital in the offline world (e.g phones).	1	12
Digital beskyttelse	Digital protection	1	1
Digital chikane	Digital harassment	1	1
Digital finansiering	Digital financing	1	1
Digital indsamling	Digital collection	1	1
Digital omstilling	Digital conversion	1	1
Digitale krænkelser	Digital violations	1	1
Digitalisering	Digitization	1	1
God tone	Good tone	1	1
Medier	Media	1	4
Ulovligt indhold	Illegal content	1	1
Diskrimination	This code/topic is about discrimination in all shapes and sizes. It is about hatecrime, sexism, equality, LGBT and so on (the list is not exhausted)	1	9
Demonstration	Demonstration	1	1
Hadforbrydelser	Hate crime	1	1
LGBT	LGBT	1	9
LGBT Asyl	LGBT asylum	1	6

LGBT Flygtninge	LGBT refugees	1	2
Ligestilling	Equality	1	2
Liv i Danmark	Life in Denmark	1	1
Racisme	Racism	1	1
Sexisme	Sexism	1	1
Event	This code is for every time there is a tweet inviting to, reminding about or simply telling about or retweeting an event, can both be an online and offline event. The sub-code list is not exhaustive	1	7
Debat	Debate	1	1
Fællesbøn	Community prayer	1	1
Flygtninge	Everything about refugees (though if it is specifically about child refugees then put it under children). This is both internal politics in DK about refugees but also how the humanitarian organizations are helping refugees around the world, and about stories about/from refugees.	1	17
Familiesammenføring	Family reunification	1	1
Flygtninge politik	Refugee politics	1	1
Flygtningelejr	Refugee camps	1	5
Hjælpe pakke	Help package	1	1
Hjælpepakke	Help package	1	1
Kvoteflygtninge	Quota refugees	1	1
Litteratur	Litterature	1	1
Frivillighed	Volunteering is about when volunteers are presented in a tweet or the organizations are trying to attract/recruit more volunteers or when they in general tweet about volunteering.	1	6
Handicap	This topic is when there are tweets about handicaps/handicapped people but not discrimination tweets (these would go under discrimination). This topic entails positive/productive tweets about handicap.	1	4
Job	Whenever they tweet about job opportunities or present a new employee in the organization	1	13
Interesse konflikt		1	1
Klima	This topic is about climate. It is about a sustainable future, a green and efficient transition, climate	1	17

	politics and climate change and more, the list is not exhaustive.		
Bæredygtig fremtid	Sustainable future	1	1
Effektiv omstilling	Effective transition	1	1
Grøn omstilling	Green transition	1	1
Grønne løfter	Green promises	1	1
Klimaaftale	Climate deal	1	2
Klimadebat	Climate debate	1	2
Klimaforandringer	Climate change	1	1
Klimahandling	Climate action	1	1
Klimakrise	Climate crisis	1	1
Klimatal	Climate numbers	1	1
Klimatilpasning	Climate adjustments	1	1
Madspild	Food waste	1	1
Rede verdenen	Saving the world	1	2
Kritik	Criticism is about when a tweet is criticizing certain actions behavior or directly criticizing politics of either the Danish government or the EU. – the list of sub-codes are not exhaustive.	1	15
Asyl	Asylum	1	2
Kritik af politik	Criticism of politics	1	8
Opholdstilladelse	Residence permit	1	2
Lande	This country topic represents whenever the tweets is about a specific country to give an overview over which countries the organizations has been bothered with in this period during corona.	1	30
Brand i Bangladesh	Fire in Bangladesh	1	1
Det Arabiske Forår	The Arab Spring	1	1
Egypten	Egypt	1	1
Eksplosion Libanon	Explosion in Lebanon	1	5
Ethiopien	Ethiopia	1	2
EU	EU	1	1
Fanget i Bosnien	Trapped in Bosnia	1	1
Fattige lande	Poor countries	1	1

Ghana	Ghana	1	1
Græske flygtningelejre	Greek refugee camps	1	1
Hongkong	Hong Kong	1	1
Indien	India	1	1
Irak	Iraq	1	1
Malawi	Malawi	1	1
Nepal	Nepal	1	2
Solidaritet i Afrika	Solidarity in Africa	1	1
Sudan	Sudan	1	1
Sydsudan	South Sudan	1	1
Syrien	Syria	1	4
Tyrkiet	Turkey	1	1
Uganda	Uganda	1	1
Unge Libanesere	Young Lebanese	1	1
Menneskerettigheder	Human rights is about rights in many relations, both regarding civil rights, activism, democracy, prison rights, human rights in general, violations and so on. The list of sub-codes are not exhaustive.	1	41
Aktivisme	Activism	1	2
Civilsamfundet	Civil society	1	9
Demokrati	Democracy	1	2
Fængsel	Prison	1	4
Fængselsvilkår	Prison terms	1	2
Menneskerettighedskrise	Human rights crisis	1	1
Menneskerettighedskrænkelser	Human rights violations	1	2
Menneskerettighedsråd	Human rights council	1	1
Mental sundhed	Mental health	1	1
Retssikkerhed	Legal security	1	3
Rettigheder	Rights	1	1
Tortur	Torture	1	3
Torturkomite	Torture comity	1	1
National Hjælp	National help relates to aid/help work that the organizations are doing nationally in Denmark, for	1	12

	example helping families that suffer from alcohol abuse, social programs, counselling and in general help to people who need it. List is not exhaustive		
Alkohol	Alcohol	1	6
Hjælp til svigtede	Help to vulnerable people	1	1
Rådgivning	Counselling	1	1
Sociale tilbud	Social programs	1	1
Nødhjælp	Emergency aid entail when the organizations tweet about the aid they are providing around the worl, especially in poor countries/areas or in relations to crisis/catastrophes and so on. This also includes both humanitarian aid, and food security. List is not exhaustive	1	31
Bidrag	Contribution	1	3
Donation	Donation	1	5
Fødevaresikkerhed	Food security	1	2
Sult	Hunger	1	4
Hjælpe i verdenen	Helping in the world	1	3
Humanitær assistance	Humanitarian assistance	1	6
Humanitær	Humanitarian	1	4
Konfliktforebyggelse	Conflict prevention	1	1
Nødhjælp i Yemen	Emergency aid in Yemen	1	1
Partnerskab	Partnership refers to when the organizations partner up with another organization to make a statement, campaign, law proposal and so on and are based on the same principal as the VNA in this project.	1	7
Dødsdom	Death penalty	1	1
Politik	Politics referrers to whenever the tweets related to a certain politic, a political suggestion, a law, taxes, legal actions, and so on (not exhaustive)	1	14
Barsel	Maternity leave	1	1
Borgerforslag	Citizen proposal	1	1
Handleplan	Action plan	1	1
Leaving No One Behind	Leaving no one behind	1	1
Lovforslag	Law proposal	1	1
Monopolister	Monopolies	1	1

Omsorgsarbejde	Care work	1	1
Rapport	Rapport	1	1
reduktionsmål	Reduction goals	1	1
Regeringen	Government	1	2
Ulighed	Inequality	1	1
Skandale	Scandal	1	1
Skat	Taxes	1	1
Skattely	Tax haven	1	1
Statistik	Statistics relates to whenever the tweets use numbers to support their argument or when they are stating that they are providing numbers/statistics about a certain topic	1	6
Tak	Thank you also entail congratulation, whenever a tweet thanks or congratulate a person or organization for their work or alike.	1	11
Tillykke	Congratulations	1	1
Udsatte	Vulnerable entails a lot of different vulnerabilities, vulnerable women, vulnerable countries, violence and vulnerable institutions (not exhaustive)	1	16
Piger-Kvinders sundhed	Girls/women's health	1	1
Sundhedsfaciliteter under angreb	Health facilities under attack	1	3
Tolke	Interpreter	1	1
Udsatte kvinder	Vulnerable women	1	1
Udsatte lande	Vulnerable countries	1	1
Vold	Violence	1	6
Partnervold	Partner violence	1	2
Udvikling	Development is about development initiatives around the world, both regarding financial, social and humanitarian initiatives	1	8
Medborgerskab	Citizenship	1	1
Religion	Religion	1	2
Udviklingsarbejde	Development work	1	1
Udviklingskrise	Development crisis	1	1
Ulandsbistand	Development assistance	1	2

Verdensmålene	The sustainable development goals (SDGs) are a topic as well though it could maybe be part of both climate, human rights, development, politics and so on. This topic relates to when the tweets directly mentions either the SDGs in general or specific SDGs	1	9

Appendix 12. Code book immersion journal

Final close codes	**Description**
Socially vulnerable	Socially vulnerable entails when the observation is about abuse in some form, loneliness and refugees (just refugees but also specific refugees such as LGBTQ refugees), young moms in poor countries and in general people who are in a vulnerable position/observations about vulnerability
Human rights	Human rights entail all kind of rights and freedom. For example, freeing captured people, torture, academic freedom, different conventions but also discrimination of different kinds. Human rights represents everytime there is an observation about legal, human, ethical break of rules/boundaries that are or should be (according to the observation) a human right, or is a human rights violation
Social media activity	Social media activity is about observation about online activity across our platforms. This can entail retweet activity, post specifications, follower patterns and so on but do also include events since these activities from the offline world is brought in as an activity on their social media platforms and the events are also often held online (due to covid-19).
Support	Support entail both financial support in the form of donations, donation encouragement and financial aid, but also includes support in the form of activism, demonstrations and support of national holidays in different countries. This category is therefor both financial, powerfull and emotionel support
Focus and agendas	Focus and agendas are a pretty broad topic that includes all the organizations different focus' such as sustainability, food, hunger and so on but also includes their rethoric choices and their communication startegys, meaining it both entail what the agenda is but also how the agenda is communicated and to whom
Conflicts	Conflicts is about when ever an observation is made about a conflict (for example conflict in Syria or Myanmar), both when they criticise actions in the conflict and when the observation is about aid for conflict areas
Characteristics of NGO's	Characteristics of NGOs entails both when observations are made about the organizations structure (who is their CEO for example) but also if the organisation characterizes it self as a big irganization, a christian organization, when the organization was established and when they are hiring new employees through job posts
Children	Children entails everytime an observation is made in relation to children, it could be child lonelyness, child refugees, bullying, children in conflict, childrens rights, childrens safety and so on. This might overlap with other codes but an observation should be coded as child when we qualitatively asses that the focus on the children is in center and the facts that they are in a conflict are is an attribute to the focus/topic/observation

Covid-19	Covid-19 relates to all post that are primarily about covid-19. This code might overlap but observations should be coded as Covid-19 when it is Covid-19 that is the center of the observation and other topics are put in relation to Covid-19. This also entails observations about vaccines, goverments handling of Covid-19, emergency aid directly linking to Covid-19 and so on
Other	Other is a category with few observations coded but a category for the few things that are still important to note and code but do not fit in to the above topics. This could end up including observations that might need their own code at some point.

Appendix. 13. Twitter Scraping

READ ME:

This notebook is created for collecting tweets for our exam project with the purpose of analysing Danish humanitarians communication during Covid-19.

As the notebook contains code for scraping twitter, **do not run *restart and run all* cells** when editing the code!

In [1]:

```
#import libaries

import tweepy
import jsonpickle
from collections import defaultdict
import json
import nltk
import pandas as pd
import re
import string
import time
import datetime
import regex
import re
import numpy as np

#NLTK

nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')

from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from nltk.stem import LancasterStemmer # Porter is used below. This is an alternative, hars
from nltk.stem import WordNetLemmatizer
from nltk import word_tokenize, pos_tag
from nltk.corpus import wordnet
```

```
[nltk_data] Downloading package stopwords to /Users/Sofie/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to /Users/Sofie/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package punkt to /Users/Sofie/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /Users/Sofie/nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
```

In [2]:

```
#importing keys for twitter api
from AppCred import CONSUMER_KEY, CONSUMER_SECRET
from AppCred import ACCESS_TOKEN, ACCESS_TOKEN_SECRET

#authentication process to access the Twitter API
auth = tweepy.OAuthHandler(CONSUMER_KEY, CONSUMER_SECRET)
auth.set_access_token(ACCESS_TOKEN, ACCESS_TOKEN_SECRET)

#pass the auth variable to the API function
api = tweepy.API(auth,
          wait_on_rate_limit=True,
          wait_on_rate_limit_notify=True)
```

User information

In [3]:

```
#list of all actors
userlist_12 = ["@PlanBornefonden",'@CARE_Danmark', '@DRC_dk','@oxfamibis','@msf_dk','@Actio
               '@redbarnetdk','@danskrodekors','@SOS_Bornebyerne','@UNICEFDK','@noedhjaelp'
              '@amnestydk', '@BornsVilkar', '@WFP_DK', '@danmissiondk', '@blaakorsdanmark',
           '@menneskeret', '@RTC_DK', '@DignityDK', '@UNDP_Danmark', '@globaltfokus',
           '@RefWelcome', '@DKIndsamling', '@Feministisk_DK', '@92_gruppen', '@ADRA_Danmark
           '@lgbt_asylum', '@MissionEast', '@AVestegnen', '@CaritasDanmark']
```

In [4]:

```
#same user data through api
user_data_12 = {}

for user in userlist_12:
    data = api.get_user(user)
    user_data_12[user] = data
```

In [5]:

```
#create dataframe for user information
actors_df = pd.DataFrame()
```

In [6]:

```
#save usernames to data frame

usernames = []
for key, value in user_data_12.items():
    usernames.append(key)

actors_df['username'] = usernames
```

In [7]:

```
#create lists for informations we want to store
status_n = []
followers_n = []
friends_n = []
location = []
bio_description = []
url = []

for key, value in user_data_12.items():

    #appending information to lists
    status_n.append(user_data_12[key]._json['statuses_count'])
    followers_n.append(user_data_12[key]._json['followers_count'])
    friends_n.append(user_data_12[key]._json['friends_count'])
    location.append(user_data_12[key]._json['location'])
    url.append(user_data_12[key]._json['url'])
    bio_description.append(user_data_12[key]._json['description'])

    print(key, ', number of status:', user_data_12[key]._json['statuses_count'],
          'followers:', user_data_12[key]._json['followers_count'], 'friends:',
          user_data_12[key]._json['friends_count'])
```

```
@PlanBornefonden , number of status: 2656 followers: 2010 friends: 1473
@CARE_Danmark , number of status: 3017 followers: 1740 friends: 761
@DRC_dk , number of status: 3313 followers: 6717 friends: 586
@oxfamibis , number of status: 5192 followers: 4711 friends: 1238
@msf_dk , number of status: 5326 followers: 5510 friends: 506
@ActionAidDK , number of status: 4236 followers: 4803 friends: 1207
@redbarnetdk , number of status: 6070 followers: 7610 friends: 1425
@danskrodekors , number of status: 4343 followers: 9461 friends: 1419
@SOS_Bornebyerne , number of status: 38 followers: 270 friends: 70
@UNICEFDK , number of status: 3779 followers: 3802 friends: 1339
@noedhjaelp , number of status: 6799 followers: 8753 friends: 1621
@amnestydk , number of status: 14690 followers: 7245 friends: 1914
@BornsVilkar , number of status: 2000 followers: 4629 friends: 879
@WFP_DK , number of status: 10283 followers: 3648 friends: 1470
@danmissiondk , number of status: 1277 followers: 979 friends: 357
@blaakorsdanmark , number of status: 972 followers: 1553 friends: 646
@menneskeret , number of status: 3237 followers: 5686 friends: 1991
@RTC_DK , number of status: 257 followers: 444 friends: 172
@DignityDK , number of status: 2709 followers: 2328 friends: 1961
@UNDP_Danmark , number of status: 7185 followers: 4368 friends: 1521
@globaltfokus , number of status: 1680 followers: 2318 friends: 405
@RefWelcome , number of status: 227 followers: 493 friends: 138
@DKIndsamling , number of status: 496 followers: 443 friends: 51
@Feministisk_DK , number of status: 1287 followers: 1324 friends: 458
@92_gruppen , number of status: 6 followers: 163 friends: 39
@ADRA_Danmark , number of status: 184 followers: 774 friends: 102
@lgbt_asylum , number of status: 201 followers: 301 friends: 416
@MissionEast , number of status: 406 followers: 178 friends: 72
@AVestegnen , number of status: 300 followers: 101 friends: 316
@CaritasDanmark , number of status: 133 followers: 73 friends: 73
```

In [8]:

```
#Add lists to dataframe in new columns

actors_df['status_n'], actors_df['followers_n'], actors_df['friends_n'], actors_df['locatio
                                                  followers_n, friends_n,
                                                  location, bio_description, url
                                                 ]
```

In [9]:

```
actors_df
```

Out[9]:

	username	status_n	followers_n	friends_n	location	bio_description
0	@PlanBornefonden	2656	2010	1473	Copenhagen, Denmark	Vi er en del af @Planglobal, som arbejder i fl...
1	@CARE_Danmark	3017	1740	761	København, Danmark	Grøn udviklings- og nødhjælpsorganisation, der...
2	@DRC_dk	3313	6717	586		Vi arbejder med at skabe varige løsninger for ...
3	@oxfamibis	5192	4711	1238	Copenhagen, Denmark	We are the Danish member of @oxfam - a global ...
4	@msf_dk	5326	5510	506	København, Danmark	Læger uden Grænser (MSF) er en int. humanitær ...
5	@ActionAidDK	4236	4803	1207	Copenhagen Denmark	We fight poverty by strenghtening human rights...
6	@redbarnetdk	6070	7610	1425	Danmark	Nyt fra Red Barnet - nyheder, holdninger og ba...
7	@danskrodekors	4343	9461	1419	Denmark	Vi er verdens største humanitære organisation....
8	@SOS_Bornebyerne	38	270	70		Vi giver forældreløse og udsatte børn en barnd...
9	@UNICEFDK	3779	3802	1339	Copenhagen, Denmark	Vi hjælper børn over hele verden med at overle...
10	@noedhjaelp	6799	8753	1621		Insights fra Folkekirkens Nødhjælps medarbejde...
11	@amnestydk	14690	7245	1914	Denmark	Vi er verdens største uafhængige organisation,...
12	@BornsVilkar	2000	4629	879		Vi arbejder for, at intet barn i Danmark svigt...
13	@WFP_DK	10283	3648	1470	Copenhagen, Marmorvej 51	FN's World Food Programme (WFP) er verdens stø...
14	@danmissiondk	1277	979	357	Hellerup	Church development, interreligious #dialogue, ...
15	@blaakorsdanmark	972	1553	646	Fredericia, Danmark	Blå Kors hjælper hjemløse og misbrugere og uds...
16	@menneskeret	3237	5686	1991	Wilders Plads 8K - Kbh K.	Institut for Menneskerettigheder - Danmarks na...

	username	status_n	followers_n	friends_n	location	bio_description
17	@RTC_DK	257	444	172	Copenhagen, Denmark	Governments must take urgent action to repatri...
18	@DignityDK	2709	2328	1961	Denmark	DIGNITY kæmper for en verden fri for tortur og...
19	@UNDP_Danmark	7185	4368	1521		FN's udviklingsprogram UNDP arbejder for at af...
20	@globaltfokus	1680	2318	405	København, Danmark	Globalt Fokus er et netværk af udviklings- og ...
21	@RefWelcome	227	493	138	Copenhagen	Humanitarian NGO. We offer legal advice and do... h
22	@DKIndsamling	496	443	51		Danmarks Indsamling er en fælles indsamling ti...
23	@Feministisk_DK	1287	1324	458	Copenhagen, Denmark	Vi forsvarer menneskerettighederne, fornyer po...
24	@92_gruppen	6	163	39	Copenhagen, Denmark	
25	@ADRA_Danmark	184	774	102		International nødhjælps- og udviklingsorganisa...
26	@lgbt_asylum	201	301	416		LGBT Asylum tilbyder social og juridisk støtte...
27	@MissionEast	406	178	72	Denmark	Mission Øst er en international hjælpeorganisa...
28	@AVestegnen	300	101	316	Vestegnen, Sjælland, Danmark	Frihedsforkæmper. \nInitiativ for at skabe opm...
29	@CaritasDanmark	133	73	73	Denmark	Caritas Danmark er Den katolske Kirkes humanit...

In [11]:

```
#uncomment to store in excel

#actors_df.to_excel('All_actors.xlsx')
```

Scraping users tweets and retweets

In [4]:

```
#Actors for scraping (first iteration) - 12 NGOs from Danmarks Indsamlingen

userlist = ["@PlanBornefonden",'@CARE_Danmark', '@DRC_dk','@oxfamibis','@msf_dk','@ActionAi
            '@danskrodekors','@SOS_Bornebyerne','@UNICEFDK','@noedhjaelp']
```

In [5]:

```
# DO NOT RUN WHEN WE HAVE COLLECTED SOME TWEETS !!!

user_collected = [] # list of users already collected
```

In [6]:

```
userlist_to_collect = [user for user in userlist if user not in user_collected]
```

In [7]:

```
userlist_to_collect
```

Out[7]:

```
['@PlanBornefonden',
 '@CARE_Danmark',
 '@DRC_dk',
 '@oxfamibis',
 '@msf_dk',
 '@ActionAidDK',
 '@redbarnetdk',
 '@danskrodekors',
 '@SOS_Bornebyerne',
 '@UNICEFDK',
 '@noedhjaelp']
```

In [8]:

```
#making a sleeper function

def sleep_15_mins():
    print("Sleeping for 15 mins ...")
    for i in range(15):
        time.sleep(60)
        print(14 - i, "minutes of sleep remaining! ;pp")
```

Loop for collecting tweets

In [10]:

```
tweets = []
startDate = datetime.datetime(2021, 5, 15, 0, 0, 0)
endDate   = datetime.datetime(2022, 1, 1, 0, 0, 0)

all_tweets = {}

for user in userlist_to_collect:
    tweetCount = 0
    # Scraping status objects from the 'user' screen name
    user_tweets = []
    print("collecting tweets for:", user)
    c = tweepy.Cursor(api.user_timeline,
                      screen_name=user,
                      tweet_mode="extended" # to get the full text from a tweet.
                     ).items()
    while True:
        try:
            status = c.next()
            # check dato between start and end date, if so, save
            date = status.created_at
            if date < endDate and date >= startDate:

                #check for tweet or retweet
                retweet_match = re.search(r'^RT @\S+: ', status.full_text)
                if retweet_match:
                    tweet   = status.retweeted_status.full_text
                    retweet = retweet_match.group(0)
                else:
                    tweet   = status.full_text
                    retweet = None # use None to signal that this is not a retweet!
                user_tweets.append({"date": date, "tweet": tweet, "retweet": retweet})

                tweetCount += 1
        except tweepy.TweepError:
            sleep_15_mins()
            continue
        except StopIteration:
            break

    all_tweets[user] = user_tweets

    print('Done collecting {} tweets for user {}'.format(tweetCount, user))
        #user_collected.extend(user) # add the user to the list of user you have collected.


#take either tweet or retweet extended
```

```
collecting tweets for: @PlanBornefonden
Sleeping for 15 mins ...
14 minutes of sleep remaining! ;pp
13 minutes of sleep remaining! ;pp
12 minutes of sleep remaining! ;pp
11 minutes of sleep remaining! ;pp
10 minutes of sleep remaining! ;pp
9 minutes of sleep remaining! ;pp
8 minutes of sleep remaining! ;pp
7 minutes of sleep remaining! ;pp
```

```
6 minutes of sleep remaining! ;pp
5 minutes of sleep remaining! ;pp
4 minutes of sleep remaining! ;pp
3 minutes of sleep remaining! ;pp
2 minutes of sleep remaining! ;pp
1 minutes of sleep remaining! ;pp
0 minutes of sleep remaining! ;pp
Sleeping for 15 mins ...
```

In [23]:

```
#save dict as json on computer
import json
def convert(date):
    if isinstance(date, datetime.datetime):
        return date.__str__()


with open('all_tweets_12.json', 'w') as outfile:
    json.dump(all_tweets_json, outfile, default=convert)
```

In [32]:

```
#inspecting tweet and information
all_tweets['@PlanBornefonden'][0]
```

Out[32]:

```
{'date': datetime.datetime(2021, 5, 14, 9, 3),
 'retweet': None,
 'tweet': '13-årige Larissa bor i Sahel-regionen, og var i 2019 tvunget til
at flygte fra hendes landsby, fordi bevæbnede mænd angreb. \n\nI dag har en
humanitær indsats givet hende muligheden for at starte på en skole i sikkerh
ed på trods af de militære uroligheder, der raser i hendes land. https://t.c
o/oDQKqVNZNt'} (https://t.co/oDQKqVNZNt'})
```

In [12]:

```
#save dict to dataframe

actor1= []
tweet1 = []
date1 = []
retweet1 = []

#insepct dictionary
for user in all_tweets:
    print(user)
    for status in all_tweets[user]:
        actor1.append(user)
        tweet1.append(status['tweet'])
        date1.append(status['date'])
        retweet1.append(status['retweet'])

tweets_df_12 = pd.DataFrame()
(tweets_df_12['actor'], tweets_df_12['tweet'],
 tweets_df_12['date'], tweets_df_12['retweet']) =  actor1, tweet1, date1, retweet1
```

```
@PlanBornefonden
@CARE_Danmark
@DRC_dk
@oxfamibis
@msf_dk
@ActionAidDK
@redbarnetdk
@danskrodekors
@SOS_Bornebyerne
@UNICEFDK
@noedhjaelp
```

In [13]:

```
print(tweets_df_12.shape) #number of tweets
tweets_df_12.head()
```

```
(163, 4)
```

Out[13]:

	actor	tweet	date	retweet
0	@PlanBornefonden	800 millioner piger og kvinder har menstruatio...	2021-05-26 06:26:00	None
1	@PlanBornefonden	Tak for en virkelig udbytterig paneldiskussion...	2021-05-25 11:33:28	None
2	@PlanBornefonden	Fondsdirektør @kimskibsted er på talerlisten o...	2021-05-25 10:31:18	RT @PDJF_dk:
3	@PlanBornefonden	@BLUETOWNwifi fortæller om deres arbejde med p...	2021-05-25 10:29:22	None
4	@PlanBornefonden	Stephan Schönemann, vicedirektør i @DanishMFA ...	2021-05-25 09:51:25	None

In [14]:

```
#convert datetime (new column)
tweets_df_12['date_convert'] = pd.to_datetime(tweets_df_12['date']).dt.normalize()
```

In [15]:

```
tweets_df_12
```

Out[15]:

	actor	tweet	date	retweet	date_convert
0	@PlanBornefonden	800 millioner piger og kvinder har menstruatio...	2021-05-26 06:26:00	None	2021-05-26
1	@PlanBornefonden	Tak for en virkelig udbytterig paneldiskussion...	2021-05-25 11:33:28	None	2021-05-25
2	@PlanBornefonden	Fondsdirektør @kimskibsted er på talerlisten o...	2021-05-25 10:31:18	RT @PDJF_dk:	2021-05-25
3	@PlanBornefonden	@BLUETOWNwifi fortæller om deres arbejde med p...	2021-05-25 10:29:22	None	2021-05-25
4	@PlanBornefonden	Stephan Schönemann, vicedirektør i @DanishMFA ...	2021-05-25 09:51:25	None	2021-05-25
...	...	...	...	...	...
158	@noedhjaelp	Generalen går sjældent på gaden - men demo mod...	2021-05-19 15:23:16	None	2021-05-19
159	@noedhjaelp	Fælles udmelding fra danske NGO'er på Gaza: Ak...	2021-05-19 08:12:42	None	2021-05-19
160	@noedhjaelp	Vores chefrådgiver @sommerane følger tæt med i...	2021-05-18 12:46:44	None	2021-05-18
161	@noedhjaelp	»Jeg er 40 år. Jeg har boet i Gaza i hele mit ...	2021-05-18 06:19:59	None	2021-05-18
162	@noedhjaelp	Meget symbolsk mål idag - en fri presse er en ...	2021-05-15 20:01:08	RT @BirgitteQ:	2021-05-15

163 rows × 5 columns

In [16]:

```
#save dataframe in excel file on computer
tweets_df_12.to_excel("tweets_12_actors_15-26maj.xlsx")
```

Collecting tweets, second iteration

In [26]:

```
new_userlist = ['@lgbt_asylum', '@MissionEast', '@AVestegnen', '@CaritasDanmark']
```

In [17]:

```
#list of actors to scrape in second round
userlist = ['@amnestydk', '@BornsVilkar', '@WFP_DK', '@danmissiondk', '@blaakorsdanmark',
            '@menneskeret', '@RTC_DK', '@DignityDK', '@UNDP_Danmark', '@globaltfokus',
            '@RefWelcome', '@DKIndsamling', '@Feministisk_DK', '@92_gruppen', '@ADRA_Danmark
            ]
```

In [27]:

```
user_collected = [] # list of users already collected
```

In [28]:

```
userlist_to_collect = [user for user in new_userlist if user not in user_collected]
```

In [31]:

```
#for loop for scraping the new actors

tweets = []
startDate = datetime.datetime(2020, 1, 1, 0, 0, 0)
endDate   = datetime.datetime(2021, 5, 28, 0, 0, 0)

all_tweets = {}

for user in userlist_to_collect:
    tweetCount = 0
    # Scraping status objects from the 'user' screen name
    user_tweets = []
    print("collecting tweets for:", user)
    c = tweepy.Cursor(api.user_timeline,
                      screen_name=user,
                      tweet_mode="extended" # to get the full text from a tweet.
                     ).items()
    while True:
        try:
            status = c.next()
            # check dato between start and end date, if so, save
            date = status.created_at
            if date < endDate and date >= startDate:

                #check for tweet or retweet
                retweet_match = re.search(r'^RT @\S+: ', status.full_text)
                if retweet_match:
                    tweet   = status.retweeted_status.full_text
                    retweet = retweet_match.group(0)
                else:
                    tweet   = status.full_text
                    retweet = None # use None to signal that this is not a retweet!
                user_tweets.append({"date": date, "tweet": tweet, "retweet": retweet})

                tweetCount += 1
        except tweepy.TweepError:
            sleep_15_mins()
            continue
        except StopIteration:
            break

    all_tweets[user] = user_tweets

    print('Done collecting {} tweets for user {}'.format(tweetCount, user))
        #user_collected.extend(user) # add the user to the list of user you have collected.


#take either tweet or retweet extended
```

```
collecting tweets for: @lgbt_asylum
Done collecting 107 tweets for user @lgbt_asylum
collecting tweets for: @MissionEast
Done collecting 101 tweets for user @MissionEast
collecting tweets for: @AVestegnen
Done collecting 294 tweets for user @AVestegnen
collecting tweets for: @CaritasDanmark
Done collecting 112 tweets for user @CaritasDanmark
```

In [32]:

```
#save dict to dataframe

actor1= []
tweet1 = []
date1 = []
retweet1 = []

#insepct dictionary
for user in all_tweets:
    print(user)
    for status in all_tweets[user]:
        actor1.append(user)
        tweet1.append(status['tweet'])
        date1.append(status['date'])
        retweet1.append(status['retweet'])

#Creating new dataframe for newly scraped actors

tweets_new_actors = pd.DataFrame()
(tweets_new_actors['actor'], tweets_new_actors['tweet'],
 tweets_new_actors['date'], tweets_new_actors['retweet']) =  actor1, tweet1, date1, retweet
```

```
@lgbt_asylum
@MissionEast
@AVestegnen
@CaritasDanmark
```

In [33]:

```
tweets_new_actors
```

Out[33]:

	actor	tweet	date	retweet
0	@lgbt_asylum	Vi ses til demo om lidt! Vi slår ring om syrer...	2021-05-19 13:25:56	None
1	@lgbt_asylum	Eid Mubarak til alle vores muslimske medlemmer...	2021-05-12 20:17:55	None
2	@lgbt_asylum	Vi præsenterer hermed det faglige notat om æn...	2021-05-12 20:13:13	RT @Beskytm_dk:
3	@lgbt_asylum	I dag eksamineres DK af FN's Menneskerettighed...	2021-05-06 06:55:15	RT @menneskeret:
4	@lgbt_asylum	Forholdene i Uganda går i den forkerte retning...	2021-05-04 07:44:30	None
...	...	...	...	...
609	@CaritasDanmark	Dolores Halpin-Bachmann har arbejdet for #Cari...	2020-02-07 10:49:47	None
610	@CaritasDanmark	Børn skal være børn - https://t.co/h5xNWh7a7U ...	2020-01-31 13:48:55	None
611	@CaritasDanmark	Today, Caritas joined @ComeceEu for a @jpeurop...	2020-01-23 14:27:29	RT @CaritasEuropa:
612	@CaritasDanmark	”Ansvaret hviler i høj grad på jeres generatio...	2020-01-21 12:00:47	None
613	@CaritasDanmark	"Faith communities have 2000 years of experien...	2020-01-12 13:23:57	RT @C2G2net:

614 rows × 4 columns

In [34]:

```
#convert datetime (new column)
tweets_new_actors['date_convert'] = pd.to_datetime(tweets_new_actors['date']).dt.normalize(
```

In [35]:

```
#save new dataframe in excel file on computer
tweets_new_actors.to_excel("tweets_4_new_actors_.xlsx")
```

Lists of friends and followers

In [3]:

```
import configparser
from tweepy import API, Cursor, OAuthHandler, TweepError
```

In [83]:

```
print('Friends:', len(users_followers_friends['@PlanBornefonden']['friends']))
print('Followers:', len(users_followers_friends['@PlanBornefonden']['followers']))
```

```
Friends: 1469
Followers: 2004
```

In [84]:

```
#loop collecting ids

#create empty dict
users_followers_friends = {}

for user in userlist_12:
    ids_friends   = []
    ids_followers = []

    print("Collecting followers for:", user) #collecting followers
    #using twitter API
    c = Cursor(api.followers_ids, screen_name = user).items()
    running = True
    while running:
        try: #try loop until API throws error, then break for 15 min
            ids_followers.append(c.next())
        except tweepy.TweepError:
            sleep_15_mins()
            continue
        except StopIteration:
            running = False
    print(f'Done collecting {len(ids_followers)} followers')

    print("collecting friends for:", user) #collect friends
    c = Cursor(api.friends_ids, screen_name=screen_name).items()
    running = True
    while running:
        try:
            ids_friends.append(c.next())
        except tweepy.TweepError:
            sleep_15_mins()
            continue
        except StopIteration:
            break
    #print every time a friends and followers list are collected
    print(f'Done collecting {len(ids_friends)} friends')
    users_followers_friends[user] = {"followers": ids_followers, "friends": ids_friends}
```

```
Collecting followers for: @PlanBornefonden
Done collecting 2004 followers
collecting friends for: @PlanBornefonden
Done collecting 1469 friends
Collecting followers for: @CARE_Danmark
Done collecting 1728 followers
collecting friends for: @CARE_Danmark
Done collecting 1469 friends
Collecting followers for: @DRC_dk
Done collecting 6708 followers
collecting friends for: @DRC_dk
Done collecting 1469 friends
Collecting followers for: @oxfamibis
Done collecting 4700 followers
collecting friends for: @oxfamibis
Done collecting 1469 friends
Collecting followers for: @msf_dk
Done collecting 5535 followers
collecting friends for: @msf_dk
Done collecting 1469 friends
```

```
Collecting followers for: @ActionAidDK
Done collecting 4795 followers
collecting friends for: @ActionAidDK
Done collecting 1469 friends
Collecting followers for: @redbarnetdk
Done collecting 7607 followers
collecting friends for: @redbarnetdk
Done collecting 1469 friends
Collecting followers for: @danskrodekors
Sleeping for 15 mins ...
14 minutes of sleep remaining! ;pp
13 minutes of sleep remaining! ;pp
12 minutes of sleep remaining! ;pp
11 minutes of sleep remaining! ;pp
10 minutes of sleep remaining! ;pp
9 minutes of sleep remaining! ;pp
8 minutes of sleep remaining! ;pp
7 minutes of sleep remaining! ;pp
6 minutes of sleep remaining! ;pp
5 minutes of sleep remaining! ;pp
4 minutes of sleep remaining! ;pp
3 minutes of sleep remaining! ;pp
2 minutes of sleep remaining! ;pp
1 minutes of sleep remaining! ;pp
0 minutes of sleep remaining! ;pp
Done collecting 9443 followers
collecting friends for: @danskrodekors
Done collecting 1469 friends
Collecting followers for: @SOS_Bornebyerne
Done collecting 269 followers
collecting friends for: @SOS_Bornebyerne
Done collecting 1469 friends
Collecting followers for: @UNICEFDK
Done collecting 3794 followers
collecting friends for: @UNICEFDK
Done collecting 1469 friends
Collecting followers for: @noedhjaelp
Done collecting 8758 followers
collecting friends for: @noedhjaelp
Done collecting 1469 friends
```

In [86]:

```
#loop collecting ids for the remaning actors

for user in userlist:

    ids_friends   = []
    ids_followers = []

    print("Collecting followers for:", user)
    c = Cursor(api.followers_ids, screen_name = user).items()
    running = True
    while running:
        try:
            ids_followers.append(c.next())
        except tweepy.TweepError:
            sleep_15_mins()
            continue
        except StopIteration:
            running = False
    print(f'Done collecting {len(ids_followers)} followers')

    print("collecting friends for:", user)
    c = Cursor(api.friends_ids, screen_name=screen_name).items()
    running = True
    while running:
        try:
            ids_friends.append(c.next())
        except tweepy.TweepError:
            sleep_15_mins()
            continue
        except StopIteration:
            break
    print(f'Done collecting {len(ids_friends)} friends')
    users_followers_friends[user] = {"followers": ids_followers, "friends": ids_friends}
```

```
Collecting followers for: @amnestydk
Done collecting 7248 followers
collecting friends for: @amnestydk
Done collecting 1469 friends
Collecting followers for: @BornsVilkar
Done collecting 4595 followers
collecting friends for: @BornsVilkar
Done collecting 1469 friends
Collecting followers for: @WFP_DK
Done collecting 3650 followers
collecting friends for: @WFP_DK
Done collecting 1469 friends
Collecting followers for: @danmissiondk
Done collecting 975 followers
collecting friends for: @danmissiondk
Done collecting 1469 friends
Collecting followers for: @blaakorsdanmark
Done collecting 1553 followers
collecting friends for: @blaakorsdanmark
Done collecting 1469 friends
Collecting followers for: @menneskeret
Done collecting 5653 followers
collecting friends for: @menneskeret
Done collecting 1469 friends
```

```
Collecting followers for: @RTC_DK
Done collecting 425 followers
collecting friends for: @RTC_DK
Done collecting 1469 friends
Collecting followers for: @DignityDK
Done collecting 2293 followers
collecting friends for: @DignityDK
Done collecting 1469 friends
Collecting followers for: @UNDP_Danmark

Rate limit reached. Sleeping for: 885


Sleeping for 15 mins ...
14 minutes of sleep remaining! ;pp
13 minutes of sleep remaining! ;pp
12 minutes of sleep remaining! ;pp
11 minutes of sleep remaining! ;pp
10 minutes of sleep remaining! ;pp
9 minutes of sleep remaining! ;pp
8 minutes of sleep remaining! ;pp
7 minutes of sleep remaining! ;pp
6 minutes of sleep remaining! ;pp
5 minutes of sleep remaining! ;pp
4 minutes of sleep remaining! ;pp
3 minutes of sleep remaining! ;pp
2 minutes of sleep remaining! ;pp
1 minutes of sleep remaining! ;pp
0 minutes of sleep remaining! ;pp
Done collecting 4366 followers
collecting friends for: @UNDP_Danmark
Done collecting 1469 friends
Collecting followers for: @globaltfokus
Done collecting 2300 followers
collecting friends for: @globaltfokus
Done collecting 1469 friends
Collecting followers for: @RefWelcome
Done collecting 479 followers
collecting friends for: @RefWelcome
Done collecting 1469 friends
Collecting followers for: @DKIndsamling
Done collecting 444 followers
collecting friends for: @DKIndsamling
Done collecting 1469 friends
Collecting followers for: @Feministisk_DK
Done collecting 1332 followers
collecting friends for: @Feministisk_DK
Done collecting 1469 friends
Collecting followers for: @92_gruppen
Done collecting 162 followers
collecting friends for: @92_gruppen
Done collecting 1469 friends
Collecting followers for: @ADRA_Danmark
Done collecting 777 followers
collecting friends for: @ADRA_Danmark
Done collecting 1469 friends
```

In [ ]:

```
actor1= []
tweet1 = []
date1 = []
retweet1 = []

#insepct dictionary
for user in all_tweets:
    print(user)
    for status in all_tweets[user]:
        actor1.append(user)
        tweet1.append(status['tweet'])
        date1.append(status['date'])
        retweet1.append(status['retweet'])

#Creating new dataframe for newly scraped actors

tweets_new_actors = pd.DataFrame()
(tweets_new_actors['actor'], tweets_new_actors['tweet'],
 tweets_new_actors['date'], tweets_new_actors['retweet']) =  actor1, tweet1, date1, retweet
```

In [96]:

```


followers = []
user_list = []

#we want data this ways
#{"Planbornefonden": {"followers": [1, 2, 3], "friends": [2, 3, 6]}, "redbarnet"}

#saving the data with the above structure
for user in users_followers_friends:
    for user_id in users_followers_friends[user]["followers"]:
        user_list.append(user)
        followers.append(user_id)

followers_df = pd.DataFrame()
followers_df['actor'], followers_df['follower'] = user_list, followers
```

In [99]:

```
#SAVE DATA FOR FRIENDS

friends = []
user_list = []

#looping throuhg user's friends, saving them
for user in users_followers_friends:
    for user_id in users_followers_friends[user]["friends"]:
        user_list.append(user)
        friends.append(user_id)

friends_df = pd.DataFrame()
friends_df['actor'], friends_df['friends'] = user_list, friends
```

In [105]:

```
#Len of datasets
print(friends_df.shape)
print(followers_df.shape)

#Save to CSV
friends_df.to_csv('friends_df.csv')
followers_df.to_csv('followers_df.csv')
```

```
(38194, 2)
(91593, 2)
```

From list of id's to usernames

In [11]:

```
#read data from computer

friends_df = pd.read_csv('friends_df.csv')
friends_df #data only contains ids, and now we want the usernames
```

Out[11]:

	Unnamed: 0	actor	friends
0	0	@PlanBornefonden	4121845336
1	1	@PlanBornefonden	1381949996606558208
2	2	@PlanBornefonden	1053010495
3	3	@PlanBornefonden	867291066839556096
4	4	@PlanBornefonden	2725300877
...	...	...	...
38189	38189	@ADRA_Danmark	95868448
38190	38190	@ADRA_Danmark	14810076
38191	38191	@ADRA_Danmark	9261402
38192	38192	@ADRA_Danmark	58569858
38193	38193	@ADRA_Danmark	41098331

38194 rows × 3 columns

In [ ]:

```
#reading the data for followers
followers_df = pd.read_csv('followers_df.csv')
followers_df
```

In [ ]:

```
#Friend id to list
friend_list = friends_df['friends'].tolist()
len(friend_list)
```

In [ ]:

```
#Followers id to list
followers_list = followers_df['follower'].tolist()
len(followers_list)
```

In [ ]:

```
import traceback
import time


#Finding @usernames for id's (friends)
ids = friend_list
info = []
for i in range(0, len(ids), 100): #taking chunks of 100
    try:
        chunk = ids[i:i+100]
        info.extend(api.lookup_users(user_ids=chunk))
    except:
        import traceback
        traceback.print_exc()
        print('Something went wrong, skipping...')
    time.sleep(10)
```

In [ ]:

```
#Saving names
data = [x._json for x in info]
df = pd.DataFrame(data)
df = df[['id', 'name', 'screen_name']]
df.to_csv('friends_name.csv', index=False)
```

In [ ]:

```
#Finding @usernames for id's (followers)

import traceback
import time
ids = followers_list
info = []
for i in range(0, len(ids), 100):
    try:
        chunk = ids[i:i+100]
        info.extend(api.lookup_users(user_ids=chunk))
    except:
        import traceback
        traceback.print_exc()
        print('Something went wrong, skipping...')
    time.sleep(10)
```

In [ ]:

```
#saving followers names
data = [x._json for x in info]
df = pd.DataFrame(data)
df = df[['id', 'name', 'screen_name']]
df.to_csv('followers_name.csv', index=False)
```

In [ ]:

```
followers_name_df = pd.read_csv('followers_name.csv')
followers_name_df.head()
```

Concatenate data frames with ID and ID names

In [ ]:

```
#Making one friends dataset by merging the actors, friends ids and friends names
friends_df1 = pd.concat([friends_df, friends_name_df], axis=1)
friends_df1.drop(['friends'], axis=1)
friends_df1.to_csv('friends_df1.csv', index=False)
```

In [ ]:

```
#Making one followers dataset by merging the actors, followers ids and followers names
followers_df1 = pd.concat([followers_df, followers_name_df], axis=1)
followers_df1
followers_df1.to_csv('followers_df1.csv', index=False)
```

Appendix 14. Data Processing of tweets

In [1]:

```
#import relevant packages

import pandas as pd
import numpy as np

from datetime import datetime #To check start and end time when running code
from tqdm import tqdm #This is for creating progress bars.
import logging #This is to provide logging of information when running the LDA
import sys #This is to disable logging when it's no longer needed
import pickle #To store and open previously saved machine learning models

#import nlkt libaries
import nltk
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from nltk.stem import LancasterStemmer # Porter is used below. This is an alternative, hars
from nltk.stem import WordNetLemmatizer
from nltk import word_tokenize, pos_tag
from nltk.corpus import wordnet

#Language detection libaries
from langdetect import detect
import fasttext

#Importing packages for data visualization
import matplotlib.pyplot as plt
import seaborn as sns

#import packages for regular expressions
import regex
import re

#Importing NLTK and NLP packages
import nltk
from nltk.tokenize import TweetTokenizer
import string
from collections import defaultdict
```

In [2]:

```
#read in data - here we have scraped data in two iterations
df_12 = pd.read_excel('tweets_12_actors_15maj.xlsx', index_col=0)
df_new = pd.read_excel('tweets_new_actors_18maj.xlsx', index_col=0)
df_new_12_new = pd.read_excel('tweets_12_actors_15-26maj.xlsx', index_col=0)
df_new_1 = pd.read_excel('tweets_new_actors_19-26maj.xlsx', index_col=0)
df_4 = pd.read_excel('tweets_4_new_actors_.xlsx', index_col=0)
```

In [3]:

```
#creating one data set
data = pd.concat([df_12, df_new, df_new_12_new,
                  df_new_1, df_4], join='inner', ignore_index=True)
data = data.reset_index(drop=True)
```

In [4]:

```
print("Total dataset:", data.shape)
```

```
Total dataset: (12176, 5)
```

In [5]:

```
# we found out that @Feministisk_DK is a political party
data = data[data.actor != '@Feministisk_DK']
print("Total dataset:", data.shape)
data.head()
```

```
Total dataset: (12139, 5)
```

Out[5]:

	actor	tweet	date	retweet	date_convert
0	@PlanBornefonden	13-årige Larissa bor i Sahel-regionen, og var ...	2021-05-14 09:03:00	NaN	2021-05-14
1	@PlanBornefonden	Vi ønsker alle muslimer en god Eid i aften! Ei...	2021-05-12 14:00:02	NaN	2021-05-12
2	@PlanBornefonden	Kom til samtalekøkken med @BosseStine og @Clau...	2021-05-12 11:58:03	RT @dorthe10:	2021-05-12
3	@PlanBornefonden	Mali, Burkina Faso og Niger - også kendt som d...	2021-05-12 10:00:02	NaN	2021-05-12
4	@PlanBornefonden	Vores seje kollega Iben Østergaard Markussen f...	2021-05-12 09:23:14	RT @dorthe10:	2021-05-12

Extracting #hashtags, @mentions and emojis

In [6]:

```
#make sure all tweets are strings
type(data.tweet[0])
data['tweet'] = data['tweet'].apply(str)
```

In [7]:

```
#Saving @ mentions in another column
mentions = []
for index, s in data.tweet.iteritems():
    results = []
    if '@' in s:
        result = re.findall("(?<![@\w])@(\w{1,25})", s)
        results.append(', '.join(result))
    else:
        results = None
    mentions.append(results)

data['@mentions'] = mentions
```

In [8]:

```
# Saving list of #hashtags in another column before cleaning text
hashtags = []
for index, s in data.tweet.iteritems():
    results = []
    if '#' in s:
        result = re.findall("(?<![@\w])#(\w{1,25})", s)
        # make for loop saving the #
        #hashtags.append(result)
        results.append(' '.join(result))
    else:
        results = ''
    hashtags.append(results)

data['#hashtags'] = hashtags
```

In [9]:

```
hashtags_1 = []
for row in hashtags:
    item = str(row)
    item = re.sub('\[', ' ', item) # only keeping letters
    item = re.sub('\]', ' ', item) # only keeping letters
    item = re.sub('\'', ' ', item) # only keeping letters
    item = re.sub(r'\s+', " ", item) #remove more whitespaces
    hashtags_1.append(item)

data['#hashtags'] = hashtags_1
```

In [10]:

```
# Saving list of emojis (😊🤔') in another column before cleaning text
emojis = []

#We extract all emojis but also all characters (which we remove below)
for index, s in data.tweet.iteritems():
        result = re.findall(r'[^\w\s,]', s)
        emojis.append(', '.join(result))

data['emojis'] = emojis
```

In [11]:

```
#we need to remove all characters from
cleaned = []

for characters in data['emojis']:
    #item = re.sub(r'@\S+', "", text) #removing @mentions
    item = regex.sub(r'\p{PUNCTUATION}', "", characters) #remove punctaion
    item = re.sub(r'\s+', " ", item) #remove more whitespaces
    cleaned.append(item)

data['emojis'] = cleaned
```

In [12]:

```
data.emojis[16]
```

Out[12]:

' 👏 '

In [13]:

```
#inspecting new agregated dataset
data.head(3)
```

Out[13]:

	actor	tweet	date	retweet	date_convert	@mentions	#hashta
0	@PlanBornefonden	13-årige Larissa bor i Sahel-regionen, og var ...	2021-05-14 09:03:00	NaN	2021-05-14	None	
1	@PlanBornefonden	Vi ønsker alle muslimer en god Eid i aften! Ei...	2021-05-12 14:00:02	NaN	2021-05-12	None	
2	@PlanBornefonden	Kom til samtalekøkken med @BosseStine og @Clau...	2021-05-12 11:58:03	RT @dorthe10:	2021-05-12	[BosseStine, ClausMeyerDK]	dkfc

Pre-processing of tweets and retweets

In [14]:

```
#Define remove emojis function
RE_EMOJI = re.compile('[\U00010000-\U0010ffff]', flags=re.UNICODE)

def strip_emoji(text):
    return RE_EMOJI.sub(r'', text)
```

In [15]:

```
#Define another remove emojis function
RE_EMOJI2 =  re.compile(u'['
    u'\U0001F300-\U0001F64F'
    u'\U0001F680-\U0001F6FF'
    u'\u2600-\u26FF\u2700-\u27BF]+',
    re.UNICODE)

def strip_emoji_2(text):
    return RE_EMOJI2.sub(r'', text)
```

In [16]:

```
def remove_emojis(data):
    emoj = re.compile("["
        u"\U0001F600-\U0001F64F"  # emoticons
        u"\U0001F300-\U0001F5FF"  # symbols & pictographs
        u"\U0001F680-\U0001F6FF"  # transport & map symbols
        u"\U0001F1E0-\U0001F1FF"  # flags (iOS)
        u"\U00002500-\U00002BEF"  # chinese char
        u"\U00002702-\U000027B0"
        u"\U00002702-\U000027B0"
        u"\U000024C2-\U0001F251"
        u"\U0001f926-\U0001f937"
        u"\U00010000-\U0010ffff"
        u"\u2640-\u2642"
        u"\u2600-\u2B55"
        u"\u200d"
        u"\u23cf"
        u"\u23e9"
        u"\u231a"
        u"\ufe0f"  # dingbats
        u"\u3030"
                      "]+", re.UNICODE)
    return re.sub(emoj, '', data)
```

In [17]:

```
#remove @ from actor column
data['actor'] = data.actor.str.replace(r'@','')
```

In [18]:

```
#cleaning tweets using regex

cleaned = []

for text in data['tweet']:
    item = re.sub(r'@\S+', "", text) #removing @mentions
    item = re.sub(r'#(\w+)', "", item) #removing hashtags
    item = re.sub(r'&amp;', "", item) #remove &

    #remove emojis and lower case
    item = re.sub(r'1⃣', "", item)
    item = re.sub(r'2⃣', "", item)
    item = re.sub(r'3⃣', "", item)
    item = re.sub(r'4⃣', "", item)
    item = re.sub(r'2⃣9⃣', "", item)
    item = strip_emoji(item) #removing emojis
    item = strip_emoji(item) #removing emojis
    item = strip_emoji_2(item) #removing emojis
    item = remove_emojis(item)
    item = re.sub(r"[┗┫━┓┏┛]","", item)
    item = re.sub(r"\u202F|\u2069|\u200d|\u2066","", item)


    item = re.sub(r'\d+', "", item) #remove digits first, otherwise we get eg. 13årige
    item = re.sub(r'(^[a-zA-Z]+$)', '', item) # only keeping letters
    item = item.lower() #lower cases

    item = regex.sub(r'\p{PUNCTUATION}', "", item) #remove punctaion
    item = re.sub(r'\s+', " ", item) #remove more whitespaces
    item = re.sub(r'http\S+', '', item) #remove urls
    item = item.rstrip()

    cleaned.append(item)

data['clean_text'] = cleaned
```

In [19]:

```
string_text = []
for index, row in data.iterrows():

    if row.actor == 'MissionEast':
        string = re.sub(r'mission øst', '', row.clean_text)
        string_text.append(string)
    else:
        string_text.append(row.clean_text)
    #item = re.sub(r'mission øst', '', words

data['clean_text'] = string_text
```

In [20]:

```
print(data.tweet[23])
data.clean_text[23]
```

```
Det er rystende at høre om den ulykkelige situation i Indien. Med bidrag til
@IndianRedCross er Danmark bl.a. med til at støtte ambulanceservice, indkøb
af beskyttelsesudstyr samt stoppe misinformation om sygdommen. Helt afgørend
e indsatser for at bekæmpe pandemien. #dkpol #dkaid
```

Out[20]:

```
'det er rystende at høre om den ulykkelige situation i indien med bidrag til
er danmark bla med til at støtte ambulanceservice indkøb af beskyttelsesudst
yr samt stoppe misinformation om sygdommen helt afgørende indsatser for at b
ekæmpe pandemien'
```

- We are keeping names, as these might refer to important actors, which we want to explore

Language Detector

In [21]:

```
import fasttext
#PRETRAINED_MODEL_PATH
fast = fasttext.load_model('/Users/Sofie/Desktop/tmp/lid.176.bin')
```

```
Warning : `load_model` does not return WordVectorModel or SupervisedModel an
y more, but a `FastText` object which is very similar.
```

In [22]:

```
# Functions for language classification

def tweet_cleaner(x):
    '''
    Cleans a str object x by:
    replacing '\n' with ' '
    replacing '#' with ''
    removing urls
    '''
    x = x.replace('\n', ' ')
    x = x.replace('#', '')
    x = re.sub(r'(https?:\/\/)(\s)*(www\.)?(\s)*((\w|\s)+\.)*([\w\-\s]+\/)*([\w\-]+)((\?)?[
    x = x.strip()
    return x

# fasttext function - language detector
def fast_detector(x):
    x = tweet_cleaner(x)
    try:
        return fast.predict(x)[0][0][-2:]
    except:
        pass
```

In [23]:

```
#Checking how the language detector works
print('Language:', fast_detector(data.tweet[117]))
```

```
Language: da
```

In [24]:

```
#adding language column
data['language'] = data["tweet"].apply(lambda x: fast_detector(x))
```

In [25]:

```
#expecting language

#most tweets with other lang than da or en are miscategorization due to emojis and links
data.groupby(data.language).count()
```

Out[25]:

language	actor	tweet	date	retweet	date_convert	@mentions	#hashtags	emojis	clean_text
af	1	1	1	0	1	1	1	1	1
ar	2	2	2	2	2	0	2	2	2
az	1	1	1	0	1	1	1	1	1
bg	1	1	1	0	1	0	1	1	1
cs	3	3	3	0	3	3	3	3	3
da	9201	9201	9201	3330	9201	5781	9201	9201	9201
de	23	23	23	8	23	17	23	23	23
ds	1	1	1	1	1	0	1	1	1
en	2589	2589	2589	2044	2589	1534	2589	2589	2589
es	13	13	13	1	13	10	13	13	13
et	1	1	1	0	1	0	1	1	1
fi	4	4	4	0	4	0	4	4	4
fr	9	9	9	7	9	3	9	9	9
ht	4	4	4	4	4	0	4	4	4
hu	2	2	2	0	2	1	2	2	2
id	2	2	2	0	2	2	2	2	2
it	2	2	2	0	2	2	2	2	2
kw	1	1	1	0	1	0	1	1	1
lb	1	1	1	0	1	1	1	1	1
ls	1	1	1	0	1	0	1	1	1
lt	1	1	1	0	1	0	1	1	1
mn	1	1	1	0	1	1	1	1	1
nl	4	4	4	2	4	1	4	4	4
no	238	238	238	57	238	133	238	238	238
pl	1	1	1	0	1	1	1	1	1
pt	2	2	2	1	2	1	2	2	2
ru	1	1	1	0	1	1	1	1	1
sv	22	22	22	8	22	13	22	22	22
tr	3	3	3	0	3	3	3	3	3
zh	4	4	4	0	4	2	4	4	4

In [26]:

```
#inspection
data.loc[data.language == 'ar']
```

Out[26]:

	actor	tweet	date	retweet	date_convert	@mentions	#hashtag
2579	ActionAidDK	شكرا لهولاء الناشطين المعتصمين أمام مبنى البر...	2021-03-17 15:04:40	RT @AAPalestine:	2021-03-17	None	
2587	ActionAidDK	تم إعطاء لقاح فيروس كورونا للإسرائليين الذين ...	2021-03-16 09:53:27	RT @AAPalestine:	2021-03-16	None	عا_للمطالبة_بلقاح_مجاني

In [27]:

```
#to english from de
english_tweets_fail_detection = [2174, 2749]

data.language[english_tweets_fail_detection[0]] = 'en'
data.language[english_tweets_fail_detection[1]] = 'en'
```

```
<ipython-input-27-f8c2eed03064>:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/
stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pand
as.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-v
ersus-a-copy)
  data.language[english_tweets_fail_detection[0]] = 'en'
<ipython-input-27-f8c2eed03064>:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/
stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pand
as.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-v
ersus-a-copy)
  data.language[english_tweets_fail_detection[1]] = 'en'
```

In [28]:

```
# remove tweets if language is not Danish or English

#where language detector is not consistent (ES, SE, DE)
tweets_remove_id = [2588, 8456, 8447, 8422, 7271, 4614, 3383, 8570, 8574, 11998]
remove_language = ['ar', 'fr', 'ht', 'pt'] #no text in ht

#We save these in objects now, so we can remove them after the retweet networks
#This is a methodological choice as the retweets are essential to understand the
#relations betweens actors through retweets, but as the text can not be used for
#PCA or other models relying on text data
```

Methodological choices regarding languages of twwets

- We sort out non-Danish tweet to increase the validity. In this way, we loose some data, but importantly, we are able to conclude on the findings setting them in relation to a national context, which we can investigate qualitatively.
- The language detector is not good enough to detect Danish and Norwegian tweets from oneanother. Most tweets detected as Norwegians tweets are actually written in Danish, and the rest uses a lot of the same words. Furhter, are there some tweets in Norwegian, the words will proberbly not be used in the PCA anyways as they appear to seldom. Therefore, we keep all norwegian tweets.

Stopwords and Lemmatizer

In [30]:

```
#Stopwords

stop_words_dk = set(stopwords.words('danish'))
stop_words_en = set(stopwords.words('english'))

# we add more stop words
danish_words = ['kan', 'så', 'få', 'se', 'ved', 'ser', 'hvordan', 'mere', 'nye', 'derfor',
                'får', 'gøre ', 'går', 'bla', 'mest', 'gør', 'stor', 'del', 'nå', 'både',
                'tæt', 'andre', 'bruge', 'dag', 'sige', 'vores', 'komme', 'siger', 'sagde',
               'ny', 'mellem', 'omkring', 'pga', 'fordi', 'gå', 'bare' , 'lidt', 'sætte',
                'of', 'on', 'the']



print("length of orignial stopword list:", len(stop_words_dk))

for word in danish_words:
    stop_words_dk.add(word)

print("length of new stopword list:", len(stop_words_dk))
```

```
length of orignial stopword list: 94
length of new stopword list: 135
```

In [31]:

```
#remove stopwords

#TOKENNIZE WORDS
data["words"] = data["clean_text"].str.split()

#REMOVE ENGLISH STOPWORDS IF LANGUAGE IS ENGLISH, ELSE REMOVE DANISH STOPWORDS
data["without_stopwords"] = [[x for x in row.words if x not in
                              (stop_words_en if row.language == 'en' else stop_words_dk)]
                             for row in data.iloc]
```

In [32]:

```
#lemmatize danish tweets
import lemmy

# Create an instance of the standalone lemmatizer.
lemmatizer = lemmy.load("da")
```

In [33]:

```
# creating a column with lemmas  (Danish)
lemmas = []

for words in data['without_stopwords']:
    lemmas_sentence = []
    for word in words:
        lemma = lemmatizer.lemmatize("", word)
        lemmas_sentence.append(lemma[0])
    lemmas.append(lemmas_sentence)


data['lemmas'] = lemmas
```

In [34]:

```
#putting words together
data['proc_text'] = data['lemmas'].apply(
                     lambda x: " ".join( x))
```

Bigrams

In [35]:

```
#We have the unigrams in the lemmas

tqdm.pandas() #Creates a progress bar. Use progress_apply instead of apply.
#Defining a function that will create bigrams
def bigrams(doc):

    bigrams = [] #Empty list to save the bigrams

    for bigram in list(nltk.bigrams(doc)):  #Creating bigrams and iterating over them
        bigrams.append("_".join(bigram))    #Connecting each bigram pair with an underscore

    return bigrams

#Creating a column with bigrams
data['bigrams'] = data.lemmas.progress_apply(lambda x: bigrams(x))
```

```
100%|██████████| 12139/12139 [00:00<00:00, 43588.81it/s]
```

In [36]:

```
# creating final data string for the words we want to include in the analyses

data['proc_text_all'] = [row.proc_text + ' ' + ' '.join(row.bigrams) + row['#hashtags']
                         for row in data.iloc] #includes, tokens, bigrams and hashtags
```

Pre-processesing retweets

In [37]:

```
#make all retweets to strings
data['retweet'] = data['retweet'].apply(str)
```

In [38]:

```
#removing RT from retweets
retweets = []
for retweet in data['retweet']:
    if retweet is 'nan':
        item = None
    else:
        item = re.sub(r'^RT ', '', retweet)
        item = re.sub(r': ', '', item)
    retweets.append(item)


data['retweet'] = retweets
```

```
<>:4: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:4: SyntaxWarning: "is" with a literal. Did you mean "=="?
<ipython-input-38-1f54a57ce3ca>:4: SyntaxWarning: "is" with a literal. Did y
ou mean "=="?
  if retweet is 'nan':
```

Making sub-dataset only for retweets network

In [39]:

```
#creating subset of data
data_retweets = data.loc[data['retweet'] != 'nan']
data_retweets = data_retweets.reset_index(drop=True)

print(data_retweets.shape)
data_retweets.head(3)
```

```
(5465, 16)
```

Out[39]:

	actor	tweet	date	retweet	date_convert	@mentions	#has
0	PlanBornefonden	Kom til samtalekøkken med @BosseStine og @Clau...	2021-05-12 11:58:03	@dorthe10	2021-05-12	[BosseStine, ClausMeyerDK]	c
1	PlanBornefonden	Vores seje kollega Iben Østergaard Markussen f...	2021-05-12 09:23:14	@dorthe10	2021-05-12	[radioloud_dk, MaternityF]	
2	PlanBornefonden	Godt at se @udviklingsmin og @JeppeKofod under...	2021-05-12 07:01:44	@anne_smith_p	2021-05-12	[udviklingsmin, JeppeKofod, DanishMFA]	

In [40]:

```
#remove @
data_retweets['retweet'] = data_retweets.retweet.str.replace(r'@','')

data_retweets = data_retweets.drop(columns=['tweet', 'date', 'date_convert','#hashtags','cl
```

In [41]:

```
data_retweets.head()
```

Out[41]:

	actor	retweet	@mentions
0	PlanBornefonden	dorthe10	[BosseStine, ClausMeyerDK]
1	PlanBornefonden	dorthe10	[radioloud_dk, MaternityF]
2	PlanBornefonden	anne_smith_p	[udviklingsmin, JeppeKofod, DanishMFA]
3	PlanBornefonden	SeeRap	None
4	PlanBornefonden	UNFPAEthiopia	[UNFPAEthiopia]

In [42]:

```
#saving to desktop
data_retweets.to_csv('data_retweets.csv', index=False)
```

Remove tweets with other languages for text analysis

- Remove **tweets_remove_id** = [2588, 8456, 8447, 8422, 7271, 4614, 3383, 8570, 8574]
- Remove **remove_language** = ['ar', 'fr', 'ht', 'pt'] #no text in ht
- Divide up Danish and English tweets

In [43]:

```
print('Shape before:', data.shape)
data_clean = data.drop(tweets_remove_id)
print('New shape after dropping first time:', data_clean.shape)
data_clean = data_clean[data_clean.language != 'ar']
print('New shape after dropping second time:', data_clean.shape)
data_clean = data_clean[data_clean.language != 'fr']
print('New shape after dropping second time:', data_clean.shape)
data_clean = data_clean[data_clean.language != 'ht']
print('New shape after dropping second time:', data_clean.shape)
data_clean = data_clean[data_clean.language != 'pt']
print('New shape after dropping second time:', data_clean.shape)
data_clean = data_clean[data_clean.proc_text != '']
print('New shape after dropping empty columns:', data_clean.shape)
data_clean = data_clean.reset_index(drop=True)
```

```
Shape before: (12139, 16)
New shape after dropping first time: (12129, 16)
New shape after dropping second time: (12127, 16)
New shape after dropping second time: (12118, 16)
New shape after dropping second time: (12114, 16)
New shape after dropping second time: (12112, 16)
New shape after dropping empty columns: (11957, 16)
```

In [44]:

```
#making danish data set
data_danish = data_clean[data_clean.language != 'en']
data_danish = data_danish.reset_index(drop=True)
data_danish.shape
```

Out[44]:

```
(9449, 16)
```

In [45]:

```
#data_danish_sub = data_danish[['actor', 'proc_text', '#hashtags']]
data_danish.to_csv('data_danish.csv', index=False)
```

In [46]:

```
#explore text where words are processed
for text in data_danish.proc_text:
    if 'læsse projekt' in text:
        print(text)
```

```
gange program officielt begynde uganda etiopien sammen konsortium fem ngo st
øtte sikre læring gennem leg barn flugte læsse projekt
true lokalsamfund sundhed fødevaresikkerhed støtte lokal produktion udstyre
ansigtsmaske sæbe redde liv samtidig øge befolkning indkomst læsse projekt
inden bygge dæmning kalobeyeu næsten umulig gro afgrøde problem snarere plad
s grøntsag læsse projekt spil vigtig rolle kris
vide fremme platform læsse projekt
```

Note on Lemmatizer:

It changes the danish word

- *læse* to *læsse*
- *tage* to *tagge*

Appendix 15. VNA & Social Network Analysis

Importing packets

In [1]:

```
import pandas as pd
import re
import numpy as np
import networkx as nx
from functools import reduce
```

Funtion to get centrality measures

In [45]:

```
#calculating the in_degree #not weighted

def network_measure(edgelist_object):
    colname = edgelist_object.columns[1]

    G_dir = nx.from_pandas_edgelist(edgelist_object, 'actor', colname, create_using=nx.DiGr
    G_dir_undirected = nx.from_pandas_edgelist(edgelist_object, 'actor', colname)

    #in-degree
    in_degree = dict(G_dir.in_degree())
    df_in_degree = pd.DataFrame(list(in_degree.items()),
                                columns = ['actor','in_degree']).sort_values(by='actor', as
    #out-degree
    out_degree = dict(G_dir.out_degree())
    df_out_degree = pd.DataFrame(list(out_degree.items()),
                                 columns = ['actor','out_degree']).sort_values(by='actor',

    #closeness centrality
    closeness = dict(nx.closeness_centrality(G_dir_undirected))
    df_closeness_degree = pd.DataFrame(list(closeness.items()),
                                       columns = ['actor', 'closeness']).sort_values(by = '
    #betweeness centrality
    betweeness = dict(nx.betweenness_centrality(G_dir_undirected))
    df_betweeness = pd.DataFrame(list(betweeness.items()),
                                       columns = ['actor', 'betweeness']).sort_values(by =

    dfs = [df_in_degree, df_out_degree, df_closeness_degree, df_betweeness]
    df_network_measures = reduce(lambda left,right: pd.merge(left,right,on='actor'), dfs)
    return df_network_measures
```

SNA - Retweet network

Importing data

In [2]:

```
#read data
data_retweets = pd.read_csv('data_retweets.csv')
data_retweets = data_retweets.drop(['@mentions'], axis=1) # We start by dropping the column

print(data_retweets.shape)
data_retweets.head(3)
```

```
(5465, 2)
```

Out[2]:

	actor	retweet
0	PlanBornefonden	dorthe10
1	PlanBornefonden	dorthe10
2	PlanBornefonden	anne_smith_p

Creating retweet network

In [3]:

```
retweet_dict = {}

#for actor, retweet in zip(data_retweets.actor, data_retweets.retweet):
for pair in zip(data_retweets['actor'], data_retweets['retweet']):

    if pair not in retweet_dict:
        retweet_dict[pair] = 1
    else:
        retweet_dict[pair] += 1
```

In [4]:

```
edgelist_df = pd.DataFrame(columns=["actor", "retweet", "weight"])

actors = []
retweets = []
weights = []

for actor, retweet in retweet_dict:
    weights.append(retweet_dict[(actor, retweet)])
    actors.append(actor)
    retweets.append(retweet)
```

In [5]:

```
edgelist_df['actor'], edgelist_df["retweet"], edgelist_df['weight'] = actors, retweets, wei
edgelist_df = edgelist_df.dropna()
```

In [6]:

```
edgelist_df
```

Out[6]:

	actor	retweet	weight
0	PlanBornefonden	dorthe10	67
1	PlanBornefonden	anne_smith_p	61
2	PlanBornefonden	SeeRap	1
3	PlanBornefonden	UNFPAEthiopia	1
4	PlanBornefonden	udviklingsmin	8
...	...	...	...
1213	CaritasDanmark	Caritaslb	1
1214	CaritasDanmark	Pontifex	2
1215	CaritasDanmark	Kerrinlinde	1
1216	CaritasDanmark	CaritasEuropa	1
1217	CaritasDanmark	C2G2net	1

1218 rows × 3 columns

In [7]:

```
# for all retweets
graph_weight = nx.from_pandas_edgelist(edgelist_df, "actor", "retweet", edge_attr='weight',
```

In [8]:

```
print(nx.info(graph_weight))
```

```
Name:
Type: DiGraph
Number of nodes: 915
Number of edges: 1218
Average in degree:   1.3311
Average out degree:   1.3311
```

In [9]:

```
#export to gephi --> Uncomment
#nx.write_gexf(graph_weight,'SNA_retweet_all.gexf')
```

Centrality scores for all retweets network

In [127]:

```
retweet_measures = network_measure(edgelist_df)
retweet_measures.sort_values(by=['in_degree'], ascending=False).head(10)
```

Out[127]:

	actor	in_degree	out_degree	closeness	betweeness
452	RasmusPrehn	13	0	0.426903	0.057325
141	DanishMFA	10	0	0.396357	0.056517
721	globaltfokus	9	44	0.428504	0.118088
896	udviklingsmin	8	0	0.366186	0.020755
848	redbarnetdk	7	66	0.446071	0.210471
688	dr2deadline	7	0	0.365746	0.009675
39	AndersLadekarl	7	0	0.350595	0.009117
134	DRC_dk	6	42	0.407854	0.088110
149	Denmark_UN	6	0	0.355504	0.013991
840	politiken	6	0	0.379568	0.011202

SNA - Small retweet network - Danish humanitarian NGO's

In [10]:

```
#if receiver is in actor list
#add to subset of data

scraped_actors = data_retweets.actor.unique()
small_retweets = pd.DataFrame()

for row in data_retweets.iloc:
    if row.retweet in scraped_actors:
        small_retweets = small_retweets.append(row)
```

In [11]:

```
#checking retweets
i = 0
for actor in data_retweets.retweet:
    if actor in scraped_actors:
        i +=1
        #print(actor)
print(i)
```

```
121
```

In [12]:

```
small_retweets.shape
```

Out[12]:

```
(121, 2)
```

In [13]:

```
retweet_dict = {}

#for actor, retweet in zip(data_retweets.actor, data_retweets.retweet):
for pair in zip(small_retweets.actor, small_retweets.retweet):

    if pair not in retweet_dict:
        retweet_dict[pair] = 1
    else:
        retweet_dict[pair] += 1

small_df = pd.DataFrame(columns=["actor", "retweet", "weight"])

actors = []
retweets = []
weights = []

for actor, retweet in retweet_dict:
    weights.append(retweet_dict[(actor, retweet)])
    actors.append(actor)
    retweets.append(retweet)

small_df['actor'], small_df['retweet'], small_df['weight'] = actors, retweets, weights

small_df
```

Out[13]:

	actor	retweet	weight
0	PlanBornefonden	globaltfokus	2
1	PlanBornefonden	msf_dk	1
2	PlanBornefonden	oxfamibis	1
3	PlanBornefonden	redbarnetdk	1
4	CARE_Danmark	globaltfokus	2
...	...	...	...
64	AVestegnen	amnestydk	7
65	AVestegnen	redbarnetdk	2
66	AVestegnen	AVestegnen	1
67	AVestegnen	BornsVilkar	1
68	CaritasDanmark	danmissiondk	1

69 rows × 3 columns

In [14]:

```
# for all retweets
graph_weight = nx.from_pandas_edgelist(small_df, "actor", "retweet", edge_attr='weight',cre
print(nx.info(graph_weight))
#export to gephi
#uncomment
#nx.write_gexf(graph_weight,'SNA_retweet_small.gexf')
```

```
Name:
Type: DiGraph
Number of nodes: 24
Number of edges: 69
Average in degree:   2.8750
Average out degree:   2.8750
```

Centrality scores for small retweet networks (only NGO's)

In [72]:

```
#calculating the in_degree #not weighted
retweet_measures_small = network_measure(small_df)
retweet_measures_small.sort_values(by=['in_degree'], ascending=False)
```

Out[72]:

	actor	in_degree	out_degree	closeness	betweeness
17	globaltfokus	9	6	0.547826	0.145210
23	redbarnetdk	7	7	0.639130	0.345925
22	oxfamibis	6	5	0.532609	0.081724
5	DRC_dk	6	6	0.532609	0.158733
19	menneskeret	4	1	0.368729	0.009505
6	DignityDK	4	5	0.467656	0.026812
10	UNDP_Danmark	3	2	0.391304	0.013768
13	amnestydk	3	4	0.479348	0.030190
12	WFP_DK	3	2	0.368729	0.004282
3	CARE_Danmark	3	3	0.445905	0.003463
9	RefWelcome	2	2	0.383478	0.007246
11	UNICEFDK	2	1	0.383478	0.000000
1	ActionAidDK	2	7	0.479348	0.096189
7	PlanBornefonden	2	4	0.479348	0.041013
16	danskrodekors	2	0	0.416824	0.000000
18	lgbt_asylum	2	3	0.355072	0.003661
20	msf_dk	2	0	0.355072	0.000000
21	noedhjaelp	2	2	0.348617	0.000000
2	BornsVilkar	2	1	0.416824	0.079051
0	AVestegnen	2	4	0.435771	0.032279
15	danmissiondk	1	0	0.043478	0.000000
8	RTC_DK	0	2	0.391304	0.000000
14	blaakorsdanmark	0	1	0.290514	0.000000
4	CaritasDanmark	0	1	0.043478	0.000000

SNA - Friends network

Importing data

In [146]:

```
#Here we start by uploading our twitter file which have all the friends of our actors in it
friends = pd.read_csv("friends_df1.csv")
friends = friends.iloc[: , 1:] #here remove the first column (which where just the index)

#As friends are the actors friends, which means that the names are users which the actors f
#This means that here, the actors will be the "sender" and the name will be the "reciever",
friends.rename(columns = {'actor' : 'actor', 'name' : 'reciever'}, inplace = True)

friends = friends[['actor', 'reciever']]
friends #here we see the data
```

Out[146]:

	actor	reciever
0	@PlanBornefonden	Andreas Schulz
1	@PlanBornefonden	World Water Week 2021
2	@PlanBornefonden	Yurdal Cicek
3	@PlanBornefonden	Andrea
4	@PlanBornefonden	Sif Klemensen
...	...	...
38189	@ADRA_Danmark	Mette Ditmer Denmark
38190	@ADRA_Danmark	YouGov
38191	@ADRA_Danmark	Natasha Friis Saxberg
38192	@ADRA_Danmark	Marianne S. Delkus
38193	@ADRA_Danmark	Jesper Marius Als

38194 rows × 2 columns

Preprocessing friends

In [147]:

```
#here we clean the column of sender, as it does have @ in front.

#first we make two list
cleaned_actor = []

for text in friends['actor']:
    item = re.sub(r'@', "", str(text)) #removing @ from sender
    cleaned_actor.append(item)

friends['actor'] = cleaned_actor

#dropping NaNs
friends.dropna()
```

Out[147]:

	actor	reciever
0	PlanBornefonden	Andreas Schulz
1	PlanBornefonden	World Water Week 2021
2	PlanBornefonden	Yurdal Cicek
3	PlanBornefonden	Andrea
4	PlanBornefonden	Sif Klemensen
...	...	...
38189	ADRA_Danmark	Mette Ditmer Denmark
38190	ADRA_Danmark	YouGov
38191	ADRA_Danmark	Natasha Friis Saxberg
38192	ADRA_Danmark	Marianne S. Delkus
38193	ADRA_Danmark	Jesper Marius Als

38194 rows × 2 columns

Creating friends network

In [148]:

```
graph_friends = nx.from_pandas_edgelist(friends, 'actor', 'reciever', create_using=nx.Multi

#export to gephi
#uncomment
#nx.write_gexf(graph_friends,'friends_network.gexf')

print(nx.info(graph_friends))
```

```
Name:
Type: MultiDiGraph
Number of nodes: 1492
Number of edges: 38194
Average in degree:  25.5992
Average out degree:  25.5992
```

Centrality scores for friends networks

In [126]:

```
#calculating the in_degree #not weighted
friends_measures = network_measure(friends)
friends_measures.sort_values(by=['out_degree'], ascending=False).head(10)
```

Out[126]:

	actor	in_degree	out_degree	closeness	betweeness
193	CARE_Danmark	0	1466	0.983509	0.037182
1477	oxfamibis	0	1466	0.983509	0.037182
1144	SOS_Bornebyerne	0	1466	0.983509	0.037182
172	BornsVilkar	0	1466	0.983509	0.037182
1458	blaakorsdanmark	0	1466	0.983509	0.037182
1459	danmissiondk	0	1466	0.983509	0.037182
1460	danskrodekors	0	1466	0.983509	0.037182
1361	UNDP_Danmark	0	1466	0.983509	0.037182
417	Feministisk_DK	0	1466	0.983509	0.037182
1083	RTC_DK	0	1466	0.983509	0.037182

SNA - Followers network

Importing data

In [149]:

```
#Here we start by uploading our twitter file which have all the followers in it.
followers = pd.read_csv("followers_df1.csv")
followers = followers.iloc[: , 1:] #here remove the first column (which where just the inde

#As followers are which users that are following our actor, the name is those who follow ou
#This means that here, the name will be the "sender" and the actors will be the "reciever",
followers.rename(columns = {'actor' : 'reciever', 'name' : 'actor'}, inplace = True)

followers = followers[['actor', 'reciever']]
followers
```

Out[149]:

	actor	reciever
0	Andreas Schulz	@PlanBornefonden
1	Victoria Ellen Gibson	@PlanBornefonden
2	DemwazelWorld	@PlanBornefonden
3	Jeppe Dillermand	@PlanBornefonden
4	Sandra Schmidt	@PlanBornefonden
...	...	...
91580	ADRA Deutschland	@ADRA_Danmark
91581	Impacto Social BR	@ADRA_Danmark
91582	ADRA Denmark	@ADRA_Danmark
91583	ADRA Norway	@ADRA_Danmark
91584	Gry Haugen	@ADRA_Danmark

91585 rows × 2 columns

In [150]:

```
#here we clean the column of sender, as it does have @ in front.

#first we make two list
cleaned_reciever = []

for text in followers['reciever']:
    item = re.sub(r'@', "", str(text)) #removing @ from reciever
    cleaned_reciever.append(item)

followers['reciever'] = cleaned_reciever

#dropping NaNs
followers.dropna()
```

Out[150]:

	actor	reciever
0	Andreas Schulz	PlanBornefonden
1	Victoria Ellen Gibson	PlanBornefonden
2	DemwazelWorld	PlanBornefonden
3	Jeppe Dillermand	PlanBornefonden
4	Sandra Schmidt	PlanBornefonden
...	...	...
91580	ADRA Deutschland	ADRA_Danmark
91581	Impacto Social BR	ADRA_Danmark
91582	ADRA Denmark	ADRA_Danmark
91583	ADRA Norway	ADRA_Danmark
91584	Gry Haugen	ADRA_Danmark

91347 rows × 2 columns

Creating followers network

In [151]:

```
graph_followers = nx.from_pandas_edgelist(followers, 'actor', 'reciever', create_using=nx.M

#export to gephi
#uncomment
#nx.write_gexf(graph_followers,'followers_network.gexf')

print(nx.info(graph_followers))
```

```
Name:
Type: MultiDiGraph
Number of nodes: 40878
Number of edges: 91585
Average in degree:   2.2404
Average out degree:   2.2404
```

Centrality scores for followers networks

In [121]:

```
in_degree = dict(graph_followers.in_degree())
df_in_degree = pd.DataFrame(list(in_degree.items()),
                                columns = ['actor','in_degree']).sort_values(by='actor', as
#out-degree
out_degree = dict(graph_followers.out_degree())
df_out_degree = pd.DataFrame(list(out_degree.items()),
                                 columns = ['actor','out_degree']).sort_values(by='actor',

dfs = [df_in_degree, df_out_degree]
df_network_measures = reduce(lambda left,right: pd.merge(left,right,on='actor'), dfs)
```

In [143]:

```
df_network_measures.sort_values(by=['in_degree'], ascending=False).head(10)
```

Out[143]:

	actor	in_degree	out_degree
37261	danskrodekors	9420	0
39039	noedhjaelp	8740	0
39265	redbarnetdk	7585	0
36743	amnestydk	7233	0
7643	DRC_dk	6693	0
38745	menneskeret	5642	0
38910	msf_dk	5520	0
743	ActionAidDK	4780	0
39120	oxfamibis	4690	0
5072	BornsVilkar	4588	0

VNA - Collaborations

Importing immersion journal observations

In [133]:

```
raw_data = pd.read_excel('VNA.xlsx')
raw_data
```

Out[133]:

	Actor	Receiver	Platform
0	Plan Børnefonden	Danmarks Indsamling	Facebook
1	Plan Børnefonden	Red Barnet	Twitter
2	Plan Børnefonden	Red Barnet	Twitter
3	Plan Børnefonden	Oxfam IBIS	Twitter
4	Plan Børnefonden	WFP Danmark	Twitter
...	...	...	...
308	Menneskeret	Amnesty	Twitter
309	Menneskeret	Amnesty	Twitter
310	Menneskeret	Amnesty	Facebook
311	Menneskeret	DIGNITY, Danmission	Facebook
312	Menneskeret	Amnesty	Facebook

313 rows × 3 columns

In [134]:

```
df = raw_data.set_index(raw_data.columns.drop('Receiver',1).tolist()).Receiver.str.split(',
df = df.rename(columns={'Actor' : 'actor'})
df
```

Out[134]:

	actor	Receiver	Platform
0	Plan Børnefonden	Danmarks Indsamling	Facebook
1	Plan Børnefonden	Red Barnet	Twitter
2	Plan Børnefonden	Red Barnet	Twitter
3	Plan Børnefonden	Oxfam IBIS	Twitter
4	Plan Børnefonden	WFP Danmark	Twitter
...	...	...	...
530	Menneskeret	Amnesty	Twitter
531	Menneskeret	Amnesty	Facebook
532	Menneskeret	DIGNITY	Facebook
533	Menneskeret	Danmission	Facebook
534	Menneskeret	Amnesty	Facebook

535 rows × 3 columns

In [144]:

```
graph_VNA = nx.from_pandas_edgelist(df, 'actor', 'Receiver', edge_attr = 'Platform', create
print(nx.info(graph_VNA))
```

```
Name:
Type: MultiDiGraph
Number of nodes: 32
Number of edges: 535
Average in degree:  16.7188
Average out degree:  16.7188
```

In [136]:

```
df_VNA_measures = network_measure(df)
df_VNA_measures.sort_values(by=['in_degree'], ascending=False)
```

Out[136]:

	actor	in_degree	out_degree	closeness	betweeness
23	Red Barnet	14	15	0.688889	0.160234
2	Amnesty	12	4	0.607843	0.138711
8	Danmarks Indsamling	12	0	0.574074	0.089635
18	Mellemfolkeligt Samvirke	9	13	0.645833	0.089151
26	Røde Kors	9	5	0.574074	0.130326
22	Plan Børnefonden	8	4	0.563636	0.099824
21	Oxfam IBIS	8	10	0.596154	0.062281
7	DIGNITY	7	9	0.574074	0.032804
10	Dansk Flygtningehjælp	7	5	0.553571	0.045063
12	Folkekirkens Nødhjælp	6	5	0.563636	0.009478
14	Globalt Fokus	6	13	0.607843	0.153680
5	Care Danmark	5	3	0.534483	0.006377
15	Kirkens Korshær	3	0	0.436620	0.010965
24	Refugees Welcome Danmark	3	7	0.492063	0.004883
17	Læger Uden Grænser	3	6	0.492063	0.012550
31	WFP Danmark	3	0	0.436620	0.002031
13	Global Action	3	0	0.413333	0.000000
9	Danmission	3	1	0.402597	0.004422
6	Caritas	2	3	0.436620	0.012497
25	Repatriate the Children Denmark	2	3	0.469697	0.000000
4	Børns Vilkår	2	2	0.436620	0.000000
30	UNICEF Danmark	2	5	0.484375	0.013011
27	SOS Børnebyerne	2	1	0.442857	0.000000
0	92-gruppen	1	0	0.382716	0.000000
20	Mission Øst	1	1	0.402597	0.000000
19	Menneskeret	1	3	0.424658	0.012075
1	Adra Danmark	1	9	0.574074	0.069143
16	LGBT Asylum	1	3	0.449275	0.000000
11	Dansk Folkehjælp	0	2	0.413333	0.000000
28	Sæt Dem Fri	0	1	0.382716	0.000000
29	UNDP Danmark	0	1	0.364706	0.000000
3	Blå kors	0	2	0.387500	0.000000

In [145]:

```
#export to gephi
#uncomment
#nx.write_gexf(graph_VNA,'VNA.gexf')
```

In [ ]:

```

```

Appendix 16. PCA

In [72]:

```
#import relevant packages
import numpy as np
import pandas as pd

#import packages for regular expressions
import regex
import re

#Importing NLTK and NLP packages
import nltk
from nltk.tokenize import TweetTokenizer
import string
from collections import defaultdict

# import for PCA
import scipy.sparse as sp
from adjustText import adjust_text
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from sklearn.feature_extraction.text import CountVectorizer

#import for vizualization
import matplotlib.pyplot as plt
import matplotlib.gridspec as gridspec
import seaborn as sns
```

In [73]:

```
#read in data
data_danish = pd.read_csv('data_danish.csv')
```

In [74]:

```
# bag of words (to ourselves - see which words are frequent)

#the 100 most used words
def get_top_n_words(corpus, n=None):
    vec = CountVectorizer().fit(corpus)
    bag_of_words = vec.transform(corpus)
    sum_words = bag_of_words.sum(axis=0)
    words_freq = [(word, sum_words[0, idx]) for word, idx in     vec.vocabulary_.items()]
    words_freq =sorted(words_freq, key = lambda x: x[1], reverse=True)
    return words_freq[:n]

frequent_words = get_top_n_words(data_danish.proc_text_all)
```

In [75]:

```
frequent_words[:10]
```

Out[75]:

```
[('dkpol', 3506),
 ('barn', 2630),
 ('dkaid', 2218),
 ('verden', 1160),
 ('danmark', 834),
 ('tak', 809),
 ('land', 785),
 ('støtte', 777),
 ('covid19', 736),
 ('menneske', 714)]
```

In [4]:

```
frequent_words2 = [pair[0] for pair in frequent_words if pair[1] >= 10]
```

In [5]:

```
len(frequent_words2)
```

Out[5]:

```
2920
```

In [6]:

```
words_high_freq = []

for string in data_danish.proc_text_all:
    xx = ""
    for word in string.split():
        if word in frequent_words2:
            xx += word + ' '
    words_high_freq.append(xx)
#data_danish.head()

data_danish['proc_freq'] = words_high_freq
```

In [7]:

```
data_danish.head()
```

Out[7]:

	actor	tweet	date	retweet	date_convert	@mentions	#hashtags
0	PlanBornefonden	13-årige Larissa bor i Sahel-regionen, og var ...	2021-05-14 09:03:00	NaN	2021-05-14 00:00:00	NaN	NaN
1	PlanBornefonden	Vi ønsker alle muslimer en god Eid i aften! Ei...	2021-05-12 14:00:02	NaN	2021-05-12 00:00:00	NaN	NaN
2	PlanBornefonden	Kom til samtalekøkken med @BosseStine og @Clau...	2021-05-12 11:58:03	@dorthe10	2021-05-12 00:00:00	['BosseStine, ClausMeyerDK']	dkfood
3	PlanBornefonden	Mali, Burkina Faso og Niger - også kendt som d...	2021-05-12 10:00:02	NaN	2021-05-12 00:00:00	NaN	NaN
4	PlanBornefonden	Vores seje kollega Iben Østergaard Markussen f...	2021-05-12 09:23:14	@dorthe10	2021-05-12 00:00:00	['radioloud_dk, MaternityF']	NaN

PCA

PCA without restricted local neighbours

Here we run the PCA on all types of words, as long as they occur in minimum 10 documents. The PCA includes word tokens, bigrams and hashtags, whereas the latter do not occurs as bigrams.

In [45]:

```
#SET VECTOR

# Instantiante the counting class
vectorizer = CountVectorizer(ngram_range=(1,1))
# Count the number of times each word type appear in each tweet
doc2type = vectorizer.fit_transform(data_danish.proc_freq)
# Restrict count to 1, i.e. we get the tweet-frequency of each word (how many tweets it app
doc2type[doc2type>0] = 1
# Get the sequence of words according to how they are counted
type_names = np.array(vectorizer.get_feature_names())
n_docs, n_types = doc2type.shape
```

In [46]:

```
doc2type.shape
```

Out[46]:

```
(9449, 2902)
```

In [47]:

```
#IDENTIFY INFREQUENCY WORDS

# if words are appearing in less than 10 docs, remove
abs_th = 10

# Calculate the tweet-frequency of each word
type_doc_freq = np.squeeze(np.asarray(np.sum(doc2type, axis=0)/n_docs))

# Identify indices of words that meet the threshold - returns boolean (True/False)
# array of length = number of words
type_mask = type_doc_freq >= abs_th / n_docs

# Identify the indices of words that meet the threshold
type_mask_idx = np.arange(n_types)[type_mask]

# Print number and share of words that meet criteria
print('{} words ({:0.0f}%) appear in at least {} tweets'.format(sum(type_mask), 100*sum(typ
```

```
2800 words (96%) appear in at least 10 tweets
```

Creating the actor term matrix

In [48]:

```
### Count the number of times each actor use each word

# list of indices that would sort by screen_name, which is the column of the NGO name
sidx = np.argsort(np.array(data_danish.actor))
# Identify unique actor names and where actor name changes in sorted list
actor_names, grp_start_idx = np.unique(np.array(data_danish.actor)[sidx], return_index=True
# Sum word count for all tweets by each actor
actor2type = np.add.reduceat(doc2type[sidx,:].toarray(), grp_start_idx)
n_actors = actor2type.shape[0]
```

In [49]:

```
### Scale the count data, we calculate frequency, and scale frequencies for each word

# Calculate how frequent each actor use each word
actor2type_frequency = actor2type/actor2type.sum(axis=1)[:, np.newaxis]

# Scale frequency within each associated word (zero mean, unit variance)
standardized_actor2type_frequency = StandardScaler().fit_transform(actor2type_frequency[:,
```

^Here we could choose to do another option, e.g. measure of association as between words (PPMI)

In [50]:

```
type_names[type_mask_idx][:100]
```

Out[50]:

```
array(['8marts', 'aalborg', 'aarhus', 'abort', 'absolut', 'absurd',
       'accelerere', 'additionel', 'adfærd', 'adgang', 'adgang_ren',
       'adgang_uddannelse', 'adgang_vaccine', 'administration',
       'adskille', 'advare', 'advarsel', 'advocacy', 'advokat', 'afbøde',
       'afghanistan', 'afgrøde', 'afgøre', 'afholde', 'afhængig',
       'aflyse', 'afrika', 'afrikansk', 'afrikansk_land', 'afskaffe',
       'afslag', 'afsløre', 'afsnit', 'afstand', 'afsted', 'afsætte',
       'aftale', 'aften', 'afvise', 'afvise_asylansøger', 'afværge',
       'agere', 'aggressiv', 'aktiv', 'aktivisme', 'aktivist',
       'aktivitet', 'aktuel', 'aktør', 'akut', 'akut_behov', 'akut_bruge',
       'al', 'alder', 'aldersdiskrimination', 'aldersgrænse', 'aldrig',
       'ale', 'alhol', 'alkohol', 'alkoholkultur', 'allerede',
       'allermest', 'alliance', 'alligevel', 'almindelig', 'alroj',
       'altafgørende', 'alternativ', 'altid', 'alting', 'altså', 'alvor',
       'alvorlig', 'alvorlig_konsekvens', 'ambassadør', 'ambition',
       'ambitiøs', 'amerikansk', 'amnesty', 'amnestys', 'analyse',
       'analysere', 'anbefale', 'anbefaling', 'anbringe', 'anbringe_barn',
       'anbringelse', 'and', 'andel', 'anden', 'ane', 'ane_lemche',
       'anerkende', 'anerkendelse', 'angribe', 'angribe_civil', 'angst',
       'anholde', 'ankomme'], dtype='<U28')
```

In [51]:

```
# See the raw counts
pd.DataFrame(actor2type[:,type_mask_idx], index = actor_names, columns = type_names[type_ma
```

Out[51]:

	8marts	aalborg	aarhus	abort	absolut	absurd	accelerere	additionel	adfæ
ADRA_Danmark	0	0	0	0	0	0	0	0	
AVestegnen	0	0	0	0	0	1	0	0	
ActionAidDK	1	1	4	0	1	0	1	0	
BornsVilkar	1	0	4	0	2	0	0	0	
CARE_Danmark	0	0	0	0	0	0	3	11	
CaritasDanmark	0	1	1	0	0	0	0	0	
DKIndsamling	0	0	0	0	0	0	0	0	
DRC_dk	1	1	0	0	0	0	0	0	
DignityDK	1	0	0	0	4	0	1	0	
MissionEast	0	0	0	0	0	0	0	0	
PlanBornefonden	7	0	0	1	0	0	0	0	
RTC_DK	0	0	0	0	0	0	0	0	
RefWelcome	0	0	0	0	0	0	0	0	
UNDP_Danmark	1	0	0	0	1	0	5	0	
UNICEFDK	1	0	0	1	1	0	2	0	
WFP_DK	4	0	0	0	0	0	2	0	
amnestydk	0	0	1	5	0	3	0	0	
blaakorsdanmark	0	1	4	0	1	0	0	0	
danmissiondk	0	0	0	0	0	0	0	0	
danskrodekors	0	3	4	0	0	1	0	0	
globaltfokus	0	0	0	0	1	0	0	2	
lgbt_asylum	0	0	1	0	0	0	0	0	
menneskeret	2	1	1	1	0	2	0	0	
msf_dk	0	0	0	0	0	0	0	0	
noedhjaelp	0	1	1	1	0	0	0	1	
oxfamibis	1	1	0	0	1	13	1	3	
redbarnetdk	1	0	1	1	1	0	1	0	

27 rows × 2800 columns

In [52]:

```
# See the standardized word frequencies
pd.DataFrame(standardized_actor2type_frequency, index = actor_names, columns = type_names[t
```

Out[52]:

	8marts	aalborg	aarhus	abort	absolut	absurd	accelerere	a
ADRA_Danmark	-0.467885	-0.429516	-0.504443	-0.340233	-0.573510	-0.406215	-0.437355	·
AVestegnen	-0.467885	-0.429516	-0.504443	-0.340233	-0.573510	2.083425	-0.437355	·
ActionAidDK	0.263704	0.243208	0.941193	-0.340233	0.667379	-0.406215	0.434008	·
BornsVilkar	-0.046434	-0.429516	0.328354	-0.340233	0.856183	-0.406215	-0.437355	·
CARE_Danmark	-0.467885	-0.429516	-0.504443	-0.340233	-0.573510	-0.406215	1.619179	
CaritasDanmark	-0.467885	4.303778	2.038435	-0.340233	-0.573510	-0.406215	-0.437355	·
DKIndsamling	-0.467885	-0.429516	-0.504443	-0.340233	-0.573510	-0.406215	-0.437355	·
DRC_dk	0.459216	0.422989	-0.504443	-0.340233	-0.573510	-0.406215	-0.437355	·
DignityDK	-0.116480	-0.429516	-0.504443	-0.340233	1.810640	-0.406215	-0.018812	·
MissionEast	-0.467885	-0.429516	-0.504443	-0.340233	-0.573510	-0.406215	-0.437355	·
PlanBornefonden	4.526276	-0.429516	-0.504443	0.588197	-0.573510	-0.406215	-0.437355	·
RTC_DK	-0.467885	-0.429516	-0.504443	-0.340233	-0.573510	-0.406215	-0.437355	·
RefWelcome	-0.467885	-0.429516	-0.504443	-0.340233	-0.573510	-0.406215	-0.437355	·
UNDP_Danmark	0.319887	-0.429516	-0.504443	-0.340233	0.762674	-0.406215	4.254047	·
UNICEFDK	0.301511	-0.429516	-0.504443	0.660999	0.731505	-0.406215	1.395432	·
WFP_DK	1.460536	-0.429516	-0.504443	-0.340233	-0.573510	-0.406215	0.711073	·
amnestydk	-0.467885	-0.429516	-0.112180	4.826310	-0.573510	2.084808	-0.437355	·
blaakorsdanmark	-0.467885	1.580083	3.814043	-0.340233	3.133345	-0.406215	-0.437355	·
danmissiondk	-0.467885	-0.429516	-0.504443	-0.340233	-0.573510	-0.406215	-0.437355	·
danskrodekors	-0.467885	1.469120	0.855569	-0.340233	-0.573510	0.313503	-0.437355	·
globaltfokus	-0.467885	-0.429516	-0.504443	-0.340233	2.384135	-0.406215	-0.437355	
lgbt_asylum	-0.467885	-0.429516	1.430002	-0.340233	-0.573510	-0.406215	-0.437355	·
menneskeret	0.590842	0.057254	-0.242934	0.348639	-0.573510	0.700906	-0.437355	·
msf_dk	-0.467885	-0.429516	-0.504443	-0.340233	-0.573510	-0.406215	-0.437355	·
noedhjaelp	-0.467885	0.232478	-0.148798	0.596615	-0.573510	-0.406215	-0.437355	
oxfamibis	-0.161851	-0.148106	-0.504443	-0.340233	-0.054429	3.754083	-0.072851	
redbarnetdk	-0.111041	-0.429516	-0.328160	0.124137	0.031754	-0.406215	-0.012333	·

27 rows × 2800 columns

Performing the PCA without restricted neighbours

In [53]:

```
## Option 2. Use the PCA implementation in Scikit-Learn:

# Instantiate the PCA class
pca = PCA(n_components=n_actors)

# Return unstandardized PC scores (i.e. U@S from SVD)
raw_PC = pca.fit_transform(standardized_actor2type_frequency)

# Get singular values
S = np.diag(pca.singular_values_)

# Get standardized PC scores
PC = raw_PC@np.linalg.inv(S)*np.sqrt(n_actors-1)

# Scale PC scores
scaled_PC = PC / np.max(abs(PC), axis=0)

# Obtain standardized term loadings (maps from PC to feature space)
L = pca.components_.T@S/np.sqrt(n_actors-1)

# Obtain word weights (maps from feature to PC space)
W = (pca.components_.T@np.linalg.inv(S))*np.sqrt(n_actors-1) # Get word weights (maps from

# Get proportion of captured variance
explained_variance = pca.explained_variance_ratio_
```

In [54]:

```
### Inspect loadings
# For unit-scaled features, they are the correlation between the standardized PCs and
#the features
# I.e. how much does each words standardized frequency correlate with each PC dimension
print(L.max())
print(L.min())
pd.DataFrame(L, index=type_names[type_mask_idx], columns=['PC{}'.format(d) for d in range(1
```

```
0.8315198426237244
-0.7542720950959446
```

Out[54]:

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8
8marts	-0.384020	-0.237834	0.011082	-0.175102	0.045295	-0.133367	0.246703	-0.069009
aalborg	-0.087927	0.347579	-0.230608	0.075942	-0.258236	-0.130693	-0.074621	0.686079
aarhus	0.110892	0.184110	-0.363340	-0.215267	-0.436503	0.109706	0.025915	0.460332
abort	0.135074	-0.293217	0.032109	0.216330	-0.009895	-0.139628	0.067624	0.096677
absolut	-0.080206	-0.277856	-0.072592	-0.191631	-0.330827	-0.125889	-0.151494	-0.117726
...	...	...	...	...	...	...	...	...
ønske	0.072876	0.066228	-0.214460	0.103936	-0.163156	0.312368	0.046973	-0.015230
østafrika	-0.325442	-0.132478	0.087583	-0.149554	0.188036	0.063639	0.139297	0.098112
øverst	-0.426733	-0.196300	-0.031740	-0.204080	0.053621	-0.150423	0.234231	-0.013372
øvrig	-0.065666	-0.087473	-0.364763	-0.391516	-0.194885	-0.117433	-0.134697	0.135747
øvrigt	-0.143010	-0.207759	0.158126	-0.014443	0.150292	0.381231	-0.352239	0.405020

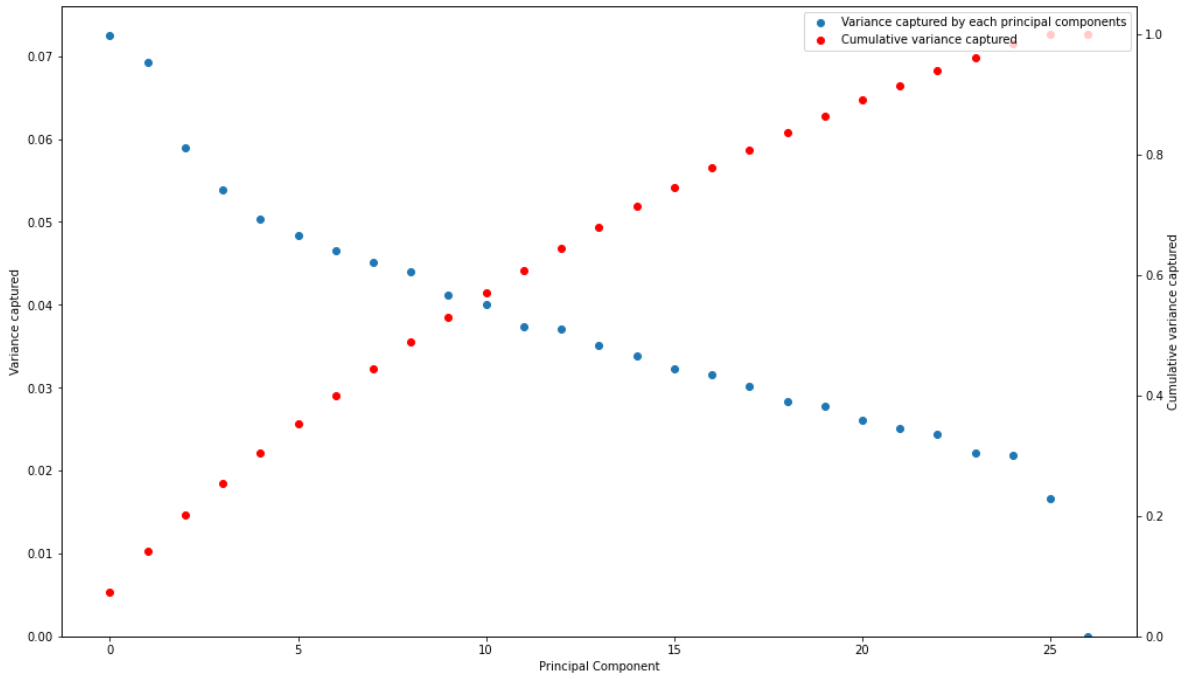
2800 rows × 27 columns

In [55]:

```
# Calculate the cumulative amount of variance captured
cumulative_variance = np.cumsum(explained_variance)
```

In [59]:

```
### Assess the amount of variance captured by each principal components

fig, ax = plt.subplots(1,1, figsize = (14,8), constrained_layout=True)

# Plot the amount of variance captured by each principal components
sc1 = ax.scatter(range(n_actors), explained_variance, label='Variance captured by each prin
ax.set_xlabel('Principal Component')
ax.set_ylabel('Variance captured')
ax.set_ylim(ymin=0)

# Plot the cumulative amount of variance captured on second y-axis
ax2 = ax.twinx()  # Create second y-axis
sc2 = ax2.scatter(range(n_actors), cumulative_variance, color='red', label='Cumulative vari
ax2.set_ylabel('Cumulative variance captured')
ax2.set_ylim(ymin=0)

plt.legend(handles = [sc1, sc2], loc=1)
plt.savefig('variance.png')
```



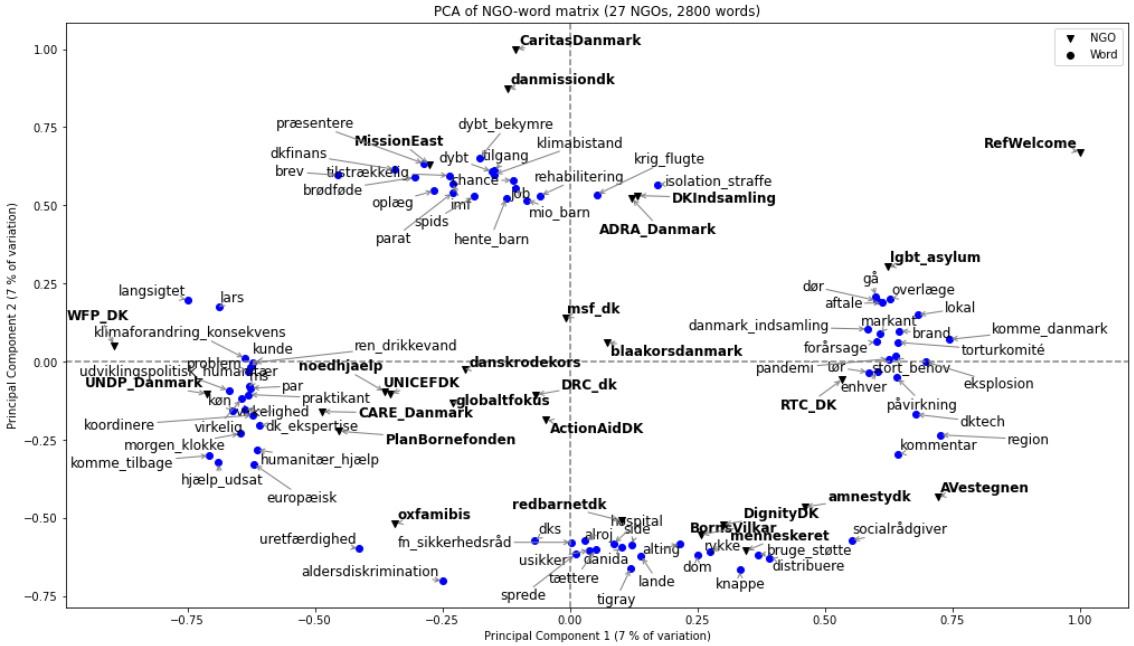
Plot dimension 1 and 2

In [57]:

```
### Identify the types that loads most on each principal dimension

# Determine how many words to plot in each direction
n_terms = 20

# Identify the indices that sorts the two first components
PCa_load_sidx = L[:,0].argsort()
PCb_load_sidx = L[:,1].argsort()

# Find the terms that load most on the first principal component
PCa_plot_idx = np.concatenate((PCa_load_sidx[:n_terms], PCa_load_sidx[-n_terms:]))

# Identify remaining indices
remain_idx = np.array([idx for idx in PCb_load_sidx if idx not in PCa_plot_idx])

# Find the remaining terms that load most on the second principal component
PCb_plot_idx = np.concatenate((remain_idx[:n_terms], remain_idx[-n_terms:]))

# Combine the indices
PC_plot_idx = np.unique(np.concatenate((PCa_plot_idx, PCb_plot_idx)))

# Get the term names
PC_plot_names = type_names[PC_plot_idx]

# Get the term loadings
PC_plot_load = L[PC_plot_idx,:]
```

In [58]:

```
### Plot the socio-symbolic constellation

fig, ax = plt.subplots(1,1, figsize = (14,8), constrained_layout=True)

# Title and label text
ax.set_title('PCA of NGO-word matrix ({} NGOs, {} words)'.format(standardized_actor2type_fr
                                                                 standardized_actor2type_fr
                                                                 fontsize = 20,fontweight='
ax.set_xlabel('Principal Component 1 ({:0.0f} % of variation)'.format(100*pca.explained_var
ax.set_ylabel('Principal Component 2 ({:0.0f} % of variation)'.format(100*pca.explained_var

# Grid to mark zero loading
ax.axvline(x=0, c='grey', linestyle='--')
ax.axhline(y=0, c='grey', linestyle='--')

# Plot standardized principal component scores
ax.scatter(scaled_PC[:,0], scaled_PC[:,1], marker = 'v', label='NGO', c='black')

# Plot word loadings
ax.scatter(PC_plot_load[:,0], PC_plot_load[:, 1], marker = 'o', label='Word', c='blue')

# Annotate the plot
texts = []
for x, y, txt in zip(scaled_PC[:,0], scaled_PC[:,1], actor_names):
    texts.append(plt.text(x, y, txt, size=12, weight='bold'))


for x, y, txt in zip(PC_plot_load[:,0], PC_plot_load[:,1], PC_plot_names):
    texts.append(plt.text(x, y, txt, size=12))

adjust_text(texts, arrowprops=dict(arrowstyle="->", color='grey')) # This part is slow

# Set legend to black
plt.legend()
leg = ax.get_legend()
leg.legendHandles[0].set_color('black')
leg.legendHandles[1].set_color('black')

plt.show()
```



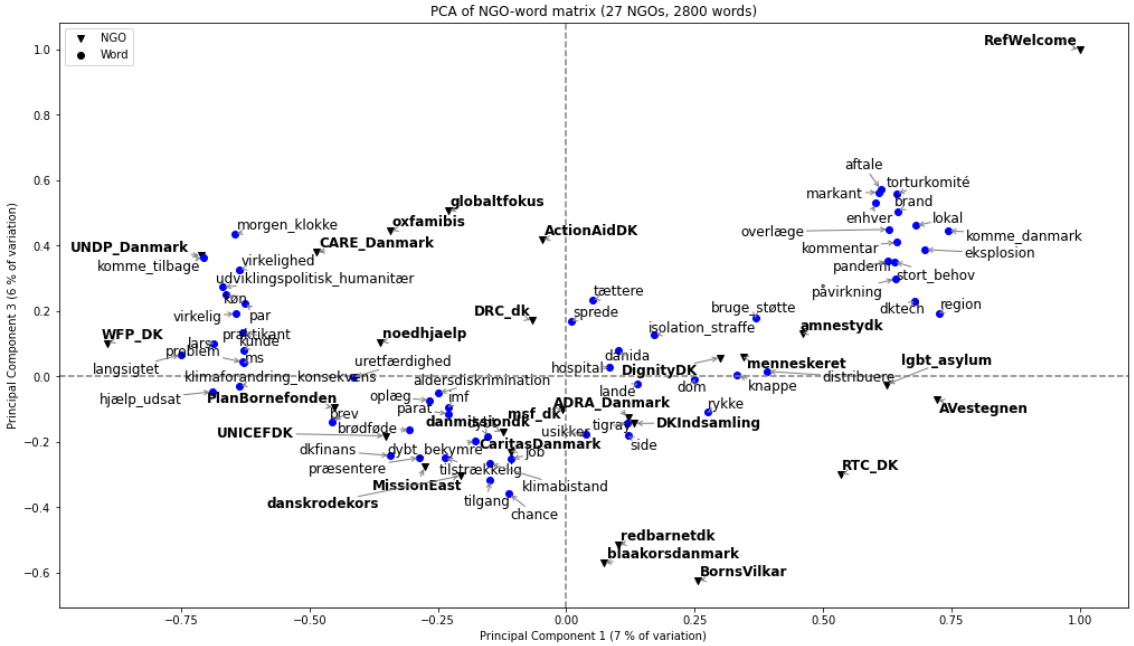
Plot dimension 3 and 4

In [60]:

```
### Identify the types that loads most on each principal dimension

# Determine how many words to plot in each direction
n_terms = 15

# Identify the indices that sorts the two first components
PCa_load_sidx = L[:,0].argsort()
PCb_load_sidx = L[:,2].argsort()

# Find the terms that load most on the first principal component
PCa_plot_idx = np.concatenate((PCa_load_sidx[:n_terms], PCa_load_sidx[-n_terms:]))

# Identify remaining indices
remain_idx = np.array([idx for idx in PCb_load_sidx if idx not in PCa_plot_idx])

# Find the remaining terms that load most on the second principal component
PCb_plot_idx = np.concatenate((remain_idx[:n_terms], remain_idx[-n_terms:]))

# Combine the indices
PC_plot_idx = np.unique(np.concatenate((PCa_plot_idx, PCb_plot_idx)))

# Get the term names
PC_plot_names = type_names[PC_plot_idx]

# Get the term loadings
PC_plot_load = L[PC_plot_idx,:]
```

In [22]:

```
#TRYING WITH DIMENSION 2 AND 3

### Plot the socio-symbolic constellation
from adjustText import adjust_text
fig, ax = plt.subplots(1,1, figsize = (14,8), constrained_layout=True)

# Title and label text
ax.set_title('PCA of NGO-word matrix ({} NGOs, {} words)'.format(standardized_actor2type_fr
                                                                 standardized_actor2type_fr
                                                                 fontsize = 20,fontweight='
ax.set_xlabel('Principal Component 1 ({:0.0f} % of variation)'.format(100*pca.explained_var
ax.set_ylabel('Principal Component 3 ({:0.0f} % of variation)'.format(100*pca.explained_var

# Grid to mark zero loading
ax.axvline(x=0, c='grey', linestyle='--')
ax.axhline(y=0, c='grey', linestyle='--')

# Plot standardized principal component scores
ax.scatter(scaled_PC[:,0], scaled_PC[:,2], marker = 'v', label='NGO', c='black')

# Plot word loadings
ax.scatter(PC_plot_load[:,0], PC_plot_load[:, 2], marker = 'o', label='Word', c='blue')

# Annotate the plot
texts = []
for x, y, txt in zip(scaled_PC[:,0], scaled_PC[:,2], actor_names):
    texts.append(plt.text(x, y, txt, size=12, weight='bold'))


for x, y, txt in zip(PC_plot_load[:,0], PC_plot_load[:,2], PC_plot_names):
    texts.append(plt.text(x, y, txt, size=12))

adjust_text(texts, arrowprops=dict(arrowstyle="->", color='grey')) # This part is slow

# Set legend to black
plt.legend()
leg = ax.get_legend()
leg.legendHandles[0].set_color('black')
leg.legendHandles[1].set_color('black')

plt.show()
```



PCA with restricted local neighbours in relation to Covid-19

In [60]:

```
#SET VECTOR

vectorizer = CountVectorizer(ngram_range=(1,1)) # Instantiante the counting class
doc2type = vectorizer.fit_transform(data_danish.proc_freq) # Count the number of times each
doc2type[doc2type>0] = 1 # Restrict count to 1, i.e. we get the tweet-frequency of each wor
type_names = np.array(vectorizer.get_feature_names()) # Get the sequence of words according
n_docs, n_types = doc2type.shape
#IDENTIFY INFREQUENCY WORDS

# if words are appearing in less than 10 docs, remove
abs_th = 10

# Calculate the tweet-frequency of each word
type_doc_freq = np.squeeze(np.asarray(np.sum(doc2type, axis=0)/n_docs))

# Identify indices of words that meet the threshold - returns boolean (True/False)
# array of length = number of words
type_mask = type_doc_freq >= abs_th / n_docs

# Identify the indices of words that meet the threshold
type_mask_idx = np.arange(n_types)[type_mask]

# Print number and share of words that meet criteria
print('{} words ({:0.0f}%) appear in at least {} tweets'.format(sum(type_mask), 100*sum(typ
```

```
2800 words (96%) appear in at least 10 tweets
```

In [61]:

```
# Calculate the binarized type co-occurence matrix
type_cooc = doc2type.T @ doc2type

# Calculate the co-occurence tweet-frequency
cooc_doc_freq = type_cooc / n_docs # Does not sum to one, as one terms occurence in a tweet
                                   # does not exclude others

# Divide observed co-occurence frequency with product of marginal word frequencies
# (expected co-occurence if 'independent')
# and scale with actual frequency, to give less weight to infrequent co-occurences
type_cooc_assoc = cooc_doc_freq.multiply(sp.csr_matrix(cooc_doc_freq /
                                                       (sp.csr_matrix(type_doc_freq).T @ sp.

# Drop self-association
type_cooc_assoc = type_cooc_assoc - sp.dia_matrix((type_cooc_assoc.diagonal()[np.newaxis, :
```

In [62]:

```
### Restrict to local neighborhood

# Define key terms
key_type = ['corona', 'covid']

# Identify all words that include the key terms as substrings (e.g.'corona' is in 'coronakr
extended_key_type = np.array([trm for trm in vectorizer.get_feature_names() if any(s_trm in
print(extended_key_type)

# Identify indices of extended key terms
ktype_idx = np.array([vectorizer.vocabulary_.get(ktype) for ktype in extended_key_type])

# Identify word associations around the extended key terms
ktype_assoc = type_cooc_assoc[ktype_idx,:]

# Get weight the extended key words according to how often they appear
extended_key_type_occurence = np.sum(doc2type[:,ktype_idx], axis=0)
extended_key_type_weight = extended_key_type_occurence / np.sum(extended_key_type_occurence

# Weight the associations and squeeze to one-dimensional array
weighted_ktype_assoc = extended_key_type_weight @ ktype_assoc
weighted_ktype_assoc = np.squeeze(np.asarray(weighted_ktype_assoc))

# Find the sequence of indices that sort the words associated with the extended key types
sidx = np.argsort(weighted_ktype_assoc)

# Remove infrequent words and the extended key types from sorted index
sidx = np.array([idx for idx in sidx if (idx in type_mask_idx) and (idx not in ktype_idx)])

# Determine size of local neighborhood; can't be greater than number of words with non-zero
n_ktype_assoc = min(400, sum(weighted_ktype_assoc[sidx]>0))
print(n_ktype_assoc)

# Select nearest (highest) associations
assoc_idx = sidx[-n_ktype_assoc:]

# Get local association matrix
assoc_matr = type_cooc_assoc[assoc_idx,:][:,assoc_idx]

# Print most associated words
print(type_names[assoc_idx])
```

```
['corona' 'coronadk' 'coronahjælp' 'coronakris' 'coronakrise'
 'coronapandemi' 'coronasmitte' 'coronavaccine' 'coronavirus'
 'coronavirusdk' 'covid' 'covid19' 'covid19dk' 'grunde_corona'
 'kamp_corona']
400
['flemming' 'interviewe' 'henvendelse' 'langsigtet' 'investere' 'beløbe'
 'ro' 'fl21' 'stopper' 'minister' 'stoppe' 'tilbage' 'sidde'
 'fælles_opråbe' 'nepal' 'reagere' 'uddele' 'donere' 'isolere' 'nogen'
 'patent' 'have' 'behandler' 'absurd' 'trivsel' 'marts' 'udgave'
 'danmark_verden' 'tjene' 'afghanistan' 'klog' 'antal' 'muligt' 'håbe'
 'eksplodere' 'fritte' 'kollega' 'hel' 'syv' 'økonomi' 'penge' 'person'
 'sidste' 'lukke' 'konstatere' 'måned' 'herhjemme' 'million_krone'
 'opgave' 'gratis' 'hør' 'aktuel' 'fremtid' 'gøre' 'flugte' 'bremse'
 'virtuel' 'made' 'klokke' 'dkmedier' 'udfordre' 'hånd' 'instagram'
 'tilgængelig' 'hjælpe_udsat' 'menneskeliv' 'påvirkning' 'ulige'
 'social_udsat' 'sf' 'uhyggelig' 'undersøgelse' 'nogensinde' 'dansker'
```

```
'indsamling' 'redde' 'tidlig' 'to' 'ms' 'bred' 'gange' 'beskytte'
'bruge_hjælp' 'mio_krone' 'stadig' 'udbrud' 'bombe' 'fem' 'vigtigt'
'imens' 'dkimpact' 'svær_tid' 'tilmelde' 'ondt' 'dag' 'første'
'politiken' 'kronprinsesse_mary' 'barn_fremtid' 'depression' 'fordoble'
'forskel' 'livsvigtig' 'bekymring' 'rød' 'desværre' 'jordan'
'sikkerhedsnet' 'trist' 'uddannelse' 'ske' 'svar' 'global_kamp' 'live'
'indslag' 'stå' 'begrænse' 'fordrive' 'følge' 'udnytte' 'angst' 'jobbe'
'hinanden' 'mia' 'dobbelt' 'pakke' 'barn_ung' 'personlig'
'udsat_menneske' 'menneske_flugte' 'genopretning' 'øge_risiko'
'mistrives' 'lave' 'tilfælde' 'sikre' 'årti' 'behov' 'presse' 'godt'
'global_solidaritet' 'skriver_generalsekretær' 'mia_krone'
'fattig_menneske' 'barn_forælder' 'international_solidaritet' 'regering'
'fantastisk' 'betyde' 'korse' 'epidemi' 'kæmpe' 'yderligere' 'online'
'fødevarehjælp' 'barn_voksen' 'restriktion' 'new' 'fællesskab'
'københavn' 'syde' 'måtte' 'akut' 'rød_korse' 'stærk' 'true' 'nødt'
'hjælpepakke' 'sanitet' 'miste' 'juridisk_chef' 'altafgørende' 'oprette'
'sej' 'tale' 'arbejder' 'sen' 'advare' 'samme' 'række' 'skærm'
'christensen' 'verden_land' 'who' 'jorden' 'lockdown' 'bank' 'gæld'
'system' 'kontinent' 'verden_barn' 'frihedsrettighed' 'million_barn'
'via' 'hungersnød' 'afbøde' 'formue' 'bekræfte' 'bange' 'fare' 'små_barn'
'frygte' 'unicef' 'johanne' 'lys' 'tydelig' 'fælles' 'udvikle' 'endnu'
'sundpol' 'hårdest' 'sygdom' 'katastrofe' 'melde' 'særligt_udsat'
'støtte' 'fokus' 'samle' 'god' 'beskytcivilsamfundet' 'ekstra_hårdt'
'sende' 'pige' 'din_barn' 'liv' 'flygtning_fordrive' 'spørgsmål'
'problem' 'normalt' 'ni' 'råd' 'allerede' 'stige' 'ulighedsrapport'
'sundhed' 'tillade' 'ulighedskrise' 'voks' 'samfund' 'hele_verden'
'indsat' 'oplyse' 'verden_udsat' 'hilse' 'påvirke' 'medicin' 'afgøre'
'tilstrækkelig' 'uddpol' 'derhjemme' 'travlt' 'forstærke' 'digital' 'små'
'isolation' 'øge' 'læsse' 'komme' 'umulig' 'snakke' 'arbejde'
'bekæmpelse' 'stor_konsekvens' 'johanne_schmidtnielsen' 'schmidtnielsen'
'sårbar_familie' 'milliard' 'inde' 'familie' 'skriver' 'holde' 'hele'
'svær' 'midt' 'nemt' 'økonomisk_kris' 'hjemme' 'ung' 'skole' 'social'
'syg' 'bor' 'forvejen' 'slut' 'kun' 'estimere' 'risiko' 'sundhedssystem'
'indsamler' 'hygiejne' 'spredning' 'sulte' 'udsat_land' 'potentiel'
'udstyre' 'men' 'gode' 'visere' 'søndag' 'virus' 'generalsekretær' 'støt'
'indsats' 'dkcivil' 'håndvaske' 'overfylde' 'rapport' 'tak'
'økonomisk_konsekvens' 'udsat_barn' 'økonomisk_system' 'sårbar' 'stor'
'sat' 'uddannelseskrise' 'kollapse' 'før' 'lille' 'ibis' 'særligt'
'global' 'blive' 'alvor' 'tid' 'skrøbelig' 'skolegang' 'oxfam_ibis'
'bruge' 'bekæmpe' 'frivillig' 'solidaritet' 'danmark'
'beskyttelsesudstyr' 'åre' 'ly' 'oxfam' 'ensomhedsstrategi' 'flere'
'hjælpenetværke' 'rig' 'flygtningelejr' 'barn_verden' 'ulighed' 'kamp'
'vaccine' 'ingen_sikre' 'sikre_før' 'før_sikre' 'fattig_land' 'forværre'
'million' 'afstand' 'afrika' 'lige' 'værre' 'menneske' 'hjælpe'
'barn_liv' 'krone' 'hjælp' 'råde' 'karantæne' 'kime' 'kris' 'ram'
'fattigdom' 'ekstra' 'mio' 'hartznvære' 'kime_hartznvære' 'dø' 'ensomhed'
'hårdt' 'risikere' 'ringe' 'sprede' 'ramme_verden' 'dksocial'
'bupartnerskab' 'teste' 'holde_afstand' 'økonomisk' 'smitte' 'land'
'konsekvens' 'verden_fattig' 'ramme' 'mobilepay' 'udsat' 'aflyse'
'gode_råde' 'pandemi' 'barn' 'grunde' 'fattig' 'ulighedsvirus' 'dkpol'
'landsindsamling' 'verden' 'dkaid']
```

Create actor term matrix

In [63]:

```
### Count the number of times each actor use each word
sidx = np.argsort(np.array(data_danish.actor)) # list of indices that would sort by screen_
actor_names, grp_start_idx = np.unique(np.array(data_danish.actor)[sidx], return_index=True
actor2type = np.add.reduceat(doc2type[sidx,:].toarray(), grp_start_idx) # Sum word count fo
n_actors = actor2type.shape[0]
```

In [64]:

```
### Scale the count data

# Calculate how frequent each actor use each word
actor2type_frequency = actor2type/actor2type.sum(axis=1)[:, np.newaxis]

# Scale frequency within each associated word (zero mean, unit variance)
standardized_actor2type_frequency = StandardScaler().fit_transform(actor2type_frequency[:,
```

In [65]:

```
# See the raw counts
pd.DataFrame(actor2type[:,assoc_idx], index = actor_names, columns = type_names[assoc_idx])
```

Out[65]:

	flemming	interviewe	henvendelse	langsigtet	investere	beløbe	ro	fl21	st
ADRA_Danmark	0	0	0	0	0	0	0	0	
AVestegnen	0	0	0	0	0	0	1	0	
ActionAidDK	0	2	0	1	6	0	0	0	
BornsVilkar	1	4	17	1	1	2	5	0	
CARE_Danmark	0	0	0	6	6	1	0	15	
CaritasDanmark	0	0	0	1	0	0	0	0	
DKIndsamling	0	0	0	0	0	0	0	0	
DRC_dk	0	1	0	5	0	3	1	0	
DignityDK	0	12	1	1	2	0	1	0	
MissionEast	1	0	1	1	0	1	1	0	
PlanBornefonden	3	1	0	8	4	1	1	0	
RTC_DK	0	6	0	0	0	0	0	0	
RefWelcome	0	1	0	0	0	0	0	0	
UNDP_Danmark	1	6	0	5	1	0	0	0	
UNICEFDK	0	3	0	3	1	4	0	0	
WFP_DK	3	1	0	5	4	0	0	0	
amnestydk	0	0	0	1	0	0	0	0	
blaakorsdanmark	0	0	0	0	0	0	0	0	
danmissiondk	0	2	0	0	0	0	0	0	
danskrodekors	3	2	0	1	0	0	2	2	
globaltfokus	1	2	0	1	1	0	0	1	
lgbt_asylum	0	2	1	0	0	0	2	0	
menneskeret	1	2	1	1	0	0	0	0	
msf_dk	0	0	0	0	0	0	0	0	
noedhjaelp	1	2	1	2	1	1	0	1	
oxfamibis	3	2	1	9	25	1	0	0	
redbarnetdk	1	1	10	9	11	0	0	2	

27 rows × 400 columns

In [66]:

```
# See the standardized word frequencies
pd.DataFrame(standardized_actor2type_frequency, index = actor_names, columns = type_names[a
```

Out[66]:

	flemming	interviewe	henvendelse	langsigtet	investere	beløbe	rc
ADRA_Danmark	-0.628817	-0.686354	-0.435693	-0.909351	-0.593521	-0.478803	-0.428966
AVestegnen	-0.628817	-0.686354	-0.435693	-0.909351	-0.593521	-0.478803	1.029452
ActionAidDK	-0.628817	-0.251616	-0.435693	-0.496292	1.803463	-0.478803	-0.428966
BornsVilkar	-0.130103	-0.185469	3.826460	-0.671398	-0.363380	0.411758	0.861879
CARE_Danmark	-0.628817	-0.686354	-0.435693	1.040395	1.292213	0.129289	-0.428966
CaritasDanmark	-0.628817	-0.686354	-0.435693	1.996934	-0.593521	-0.478803	-0.428966
DKIndsamling	-0.628817	-0.686354	-0.435693	-0.909351	-0.593521	-0.478803	-0.428966
DRC_dk	-0.628817	-0.410894	-0.435693	1.707876	-0.593521	2.459762	0.138950
DignityDK	-0.628817	0.566555	-0.226648	-0.710946	-0.209739	-0.478803	-0.213706
MissionEast	3.180435	-0.686354	1.479302	0.908168	-0.593521	2.922314	1.542965
PlanBornefonden	1.903923	-0.474374	-0.435693	2.313191	0.964851	0.274988	0.008074
RTC_DK	-0.628817	3.251821	-0.435693	-0.909351	-0.593521	-0.478803	-0.428966
RefWelcome	-0.628817	1.154686	-0.435693	-0.909351	-0.593521	-0.478803	-0.428966
UNDP_Danmark	0.303375	0.718018	-0.435693	1.314547	-0.163344	-0.478803	-0.428966
UNICEFDK	-0.628817	-0.000548	-0.435693	0.393862	-0.173378	2.772794	-0.428966
WFP_DK	1.082648	-0.543111	-0.435693	0.451643	0.459528	-0.478803	-0.428966
amnestydk	-0.628817	-0.686354	-0.435693	-0.461029	-0.593521	-0.478803	-0.428966
blaakorsdanmark	-0.628817	-0.686354	-0.435693	-0.909351	-0.593521	-0.478803	-0.428966
danmissiondk	-0.628817	2.354218	-0.435693	-0.909351	-0.593521	-0.478803	-0.428966
danskrodekors	1.814485	-0.277365	-0.435693	-0.520758	-0.593521	-0.478803	0.414248
globaltfokus	1.434592	0.349839	-0.435693	0.075169	0.358677	-0.478803	-0.428966
lgbt_asylum	-0.628817	1.640586	1.893780	-0.909351	-0.593521	-0.478803	4.368498
menneskeret	-0.002407	-0.371786	-0.120783	-0.610470	-0.593521	-0.478803	-0.428966
msf_dk	-0.628817	-0.686354	-0.435693	-0.909351	-0.593521	-0.478803	-0.428966
noedhjaelp	0.223084	-0.258550	-0.007424	-0.096411	-0.200396	0.281823	-0.428966
oxfamibis	0.457598	-0.504497	-0.253638	0.645743	3.584363	-0.155465	-0.428966
redbarnetdk	-0.206553	-0.580328	1.687121	0.903933	1.549954	-0.478803	-0.428966

27 rows × 400 columns

Creating the PCA for words restricted on Covid

In [67]:

```
## Option 2. Use the PCA implementation in Scikit-Learn:
from sklearn.decomposition import PCA

# Instantiate the PCA class
pca = PCA(n_components=n_actors)

# Return unstandardized PC scores (i.e. U@S from SVD)
raw_PC = pca.fit_transform(standardized_actor2type_frequency)

# Get singular values
S = np.diag(pca.singular_values_)

# Get standardized PC scores
PC = raw_PC@np.linalg.inv(S)*np.sqrt(n_actors-1)

# Scale PC scores
scaled_PC = PC / np.max(abs(PC), axis=0)

# Obtain standardized term loadings (maps from PC to feature space)
L = pca.components_.T@S/np.sqrt(n_actors-1)

# Obtain word weights (maps from feature to PC space)
W = (pca.components_.T@np.linalg.inv(S))*np.sqrt(n_actors-1) # Get word weights (maps from

# Get proportion of captured variance
explained_variance = pca.explained_variance_ratio_
```

In [68]:

```
### Example of relations between original data and PC scores
# NB: PCs are only unique up to a constant, so the sign (direction) can change

## From original data to PC scores
# The inner product (sumproduct) of the third NGOs standardized word frequencies and the wo
print(sum(standardized_actor2type_frequency[2,:]*W[:,1]))
# The second PC score of the third NGO
print(PC[2,1])

## From PC scores to original data
# The inner product (sumproduct) of the third NGOs PC scores and the loadings of the fourth
print(sum(PC[2,:]*L[3,:]))
# The fourth standardized word frequency of the third NGO
print(standardized_actor2type_frequency[2,3])
```

```
-0.2560932899375548
-0.2560932899375556
-0.4962924089139739
-0.4962924089139682
```

In [69]:

```
### Inspect loadings
# For unit-scaled features, they are the correlation between the standardized PCs and the f
# I.e. how much does each words standardized frequency correlate with each PC dimension
print(L.max())
print(L.min())
pd.DataFrame(L, index=type_names[assoc_idx], columns=['PC{}'.format(d) for d in range(1,L.s
```

```
0.8465155283852122
-0.6930595377097685
```

Out[69]:

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	
flemming	0.636019	0.123326	0.202055	-0.227385	0.080089	-0.056746	-0.092477	-0.
interviewe	-0.283542	0.089589	-0.106247	0.308156	0.451886	-0.231500	0.267982	0.
henvendelse	0.088426	0.038611	0.387977	0.099664	-0.014845	-0.480274	-0.088497	-0.
langsigtet	0.578582	-0.085988	-0.163679	-0.065345	-0.137697	0.368234	0.136305	-0.
investere	0.345684	-0.420531	-0.545557	-0.269684	0.042217	-0.192191	-0.151573	-0.
...	...	...	...	...	...	...	...	
ulighedsvirus	0.251923	-0.297587	-0.495239	-0.213408	0.120238	-0.151919	-0.367885	-0.
dkpol	-0.247373	-0.253298	-0.184268	-0.156624	0.375700	-0.219360	-0.301452	0.
landsindsamling	0.112261	-0.206655	0.143857	-0.068750	0.018174	0.080323	-0.075614	-0.
verden	0.476640	-0.156837	-0.460231	-0.257780	-0.176055	0.287099	0.275679	-0.
dkaid	0.558450	0.388446	-0.052651	-0.088433	0.260487	0.320323	-0.099576	0.

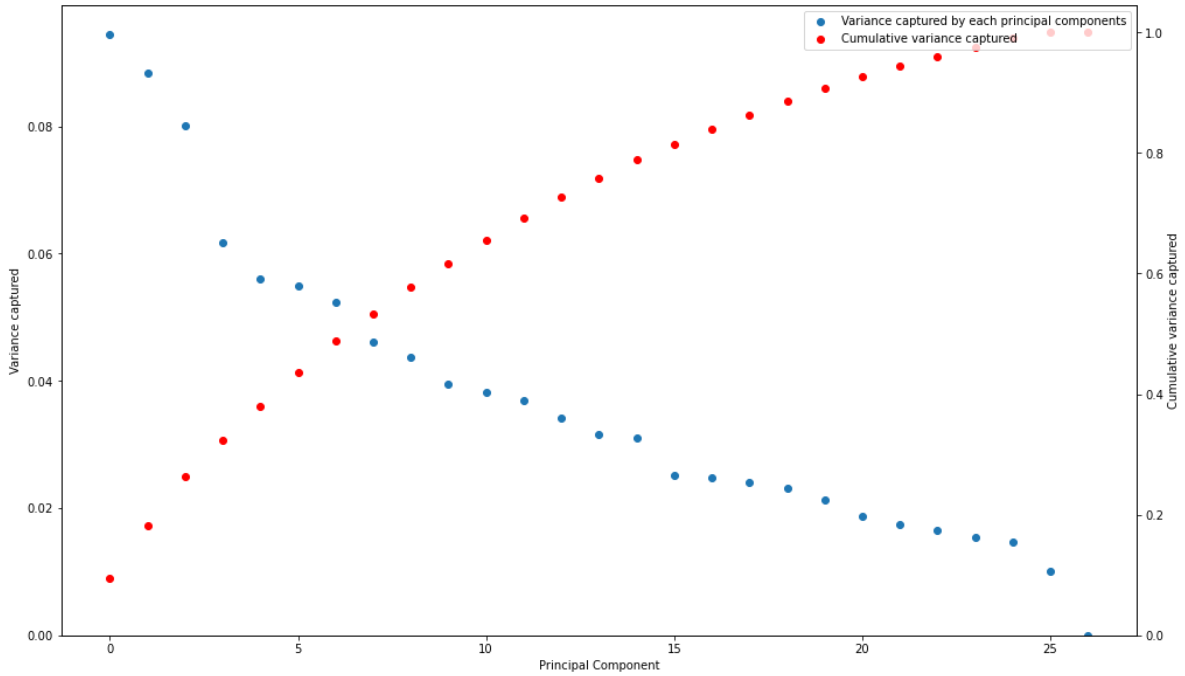
400 rows × 27 columns

In [70]:

```
# Calculate the cumulative amount of variance captured
cumulative_variance = np.cumsum(explained_variance)
```

In [71]:

```
### Assess the amount of variance captured by each principal components

fig, ax = plt.subplots(1,1, figsize = (14,8), constrained_layout=True)

# Plot the amount of variance captured by each principal components
sc1 = ax.scatter(range(n_actors), explained_variance, label='Variance captured by each prin
ax.set_xlabel('Principal Component')
ax.set_ylabel('Variance captured')
ax.set_ylim(ymin=0)

# Plot the cumulative amount of variance captured on second y-axis
ax2 = ax.twinx()  # Create second y-axis
sc2 = ax2.scatter(range(n_actors), cumulative_variance, color='red', label='Cumulative vari
ax2.set_ylabel('Cumulative variance captured')
ax2.set_ylim(ymin=0)

plt.legend(handles = [sc1, sc2], loc=1)
plt.savefig('variance_corona.png')
```

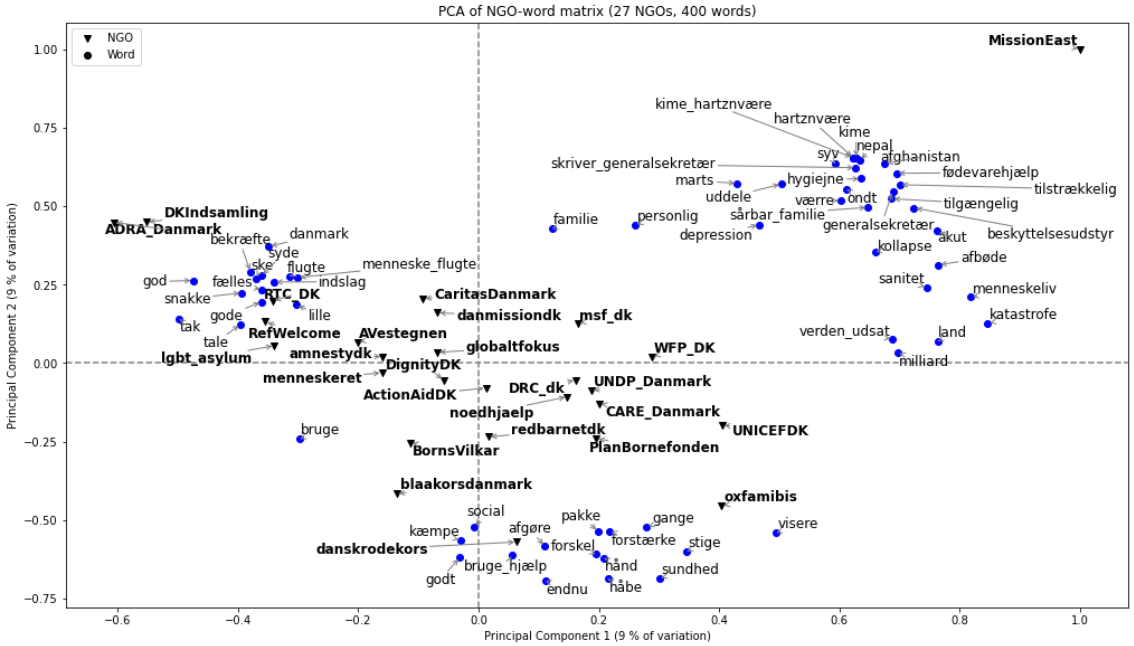


In [35]:

```
### Identify the types that loads most on each principal dimension

# Determine how many words to plot in each direction
n_terms = 15

# Identify the indices that sorts the two first components
PCa_load_sidx = L[:,0].argsort()
PCb_load_sidx = L[:,1].argsort()

# Find the terms that load most on the first principal component
PCa_plot_idx = np.concatenate((PCa_load_sidx[:n_terms], PCa_load_sidx[-n_terms:]))

# Identify remaining indices
remain_idx = np.array([idx for idx in PCb_load_sidx if idx not in PCa_plot_idx])

# Find the remaining terms that load most on the second principal component
PCb_plot_idx = np.concatenate((remain_idx[:n_terms], remain_idx[-n_terms:]))

# Combine the indices
PC_plot_idx = np.unique(np.concatenate((PCa_plot_idx, PCb_plot_idx)))

# Get the term names
PC_plot_names = type_names[assoc_idx][PC_plot_idx]

# Get the term loadings
PC_plot_load = L[PC_plot_idx,:]
```

In [36]:

```
### Plot the socio-symbolic constellation
from adjustText import adjust_text
fig, ax = plt.subplots(1,1, figsize = (14,8), constrained_layout=True)

# Title and label text
ax.set_title('PCA of NGO-word matrix ({} NGOs, {} words)'.format(standardized_actor2type_fr
                                                                 standardized_actor2type_fr
                                                                 fontsize = 20,fontweight='
ax.set_xlabel('Principal Component 1 ({:0.0f} % of variation)'.format(100*pca.explained_var
ax.set_ylabel('Principal Component 2 ({:0.0f} % of variation)'.format(100*pca.explained_var

# Grid to mark zero loading
ax.axvline(x=0, c='grey', linestyle='--')
ax.axhline(y=0, c='grey', linestyle='--')

# Plot standardized principal component scores
ax.scatter(scaled_PC[:,0], scaled_PC[:,1], marker = 'v', label='NGO', c='black')

# Plot word loadings
ax.scatter(PC_plot_load[:,0], PC_plot_load[:, 1], marker = 'o', label='Word', c='blue')

# Annotate the plot
texts = []
for x, y, txt in zip(scaled_PC[:,0], scaled_PC[:,1], actor_names):
    texts.append(plt.text(x, y, txt, size=12, weight='bold'))


for x, y, txt in zip(PC_plot_load[:,0], PC_plot_load[:,1], PC_plot_names):
    texts.append(plt.text(x, y, txt, size=12))

adjust_text(texts, arrowprops=dict(arrowstyle="->", color='grey')) # This part is slow

# Set legend to black
plt.legend()
leg = ax.get_legend()
leg.legendHandles[0].set_color('black')
leg.legendHandles[1].set_color('black')

plt.show()
```



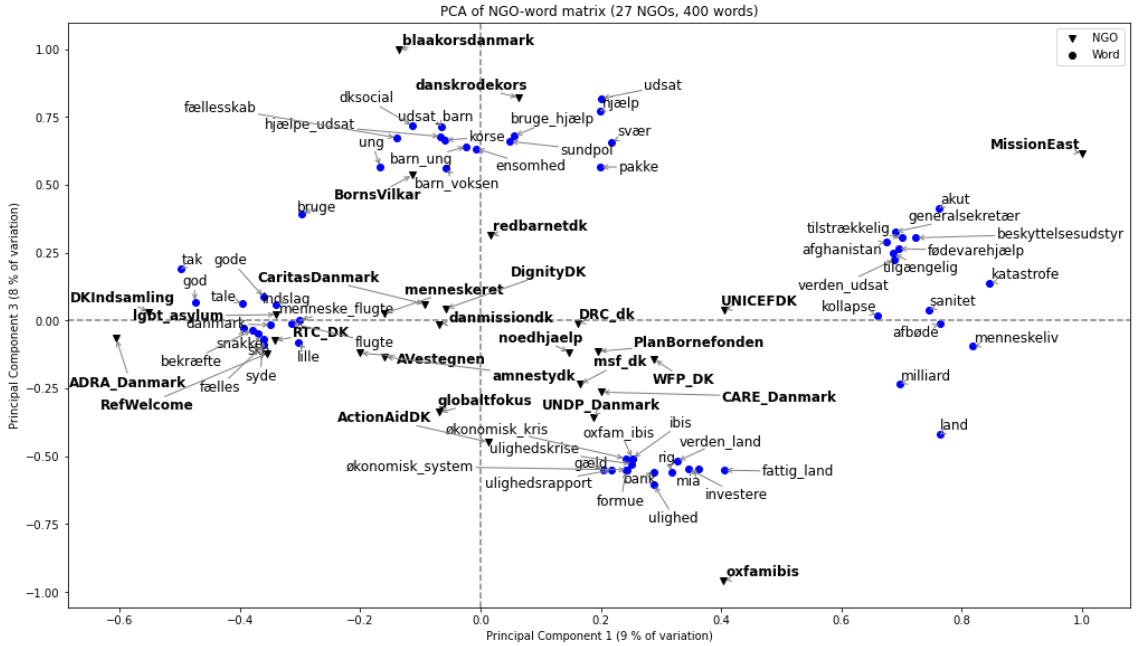
The third and fourth dimension resctricted on covid/corona with hashtags

In [41]:

```
### Identify the types that loads most on each principal dimension

# Determine how many words to plot in each direction
n_terms = 15

# Identify the indices that sorts the two first components
PCa_load_sidx = L[:,0].argsort()
PCb_load_sidx = L[:,2].argsort()

# Find the terms that load most on the first principal component
PCa_plot_idx = np.concatenate((PCa_load_sidx[:n_terms], PCa_load_sidx[-n_terms:]))

# Identify remaining indices
remain_idx = np.array([idx for idx in PCb_load_sidx if idx not in PCa_plot_idx])

# Find the remaining terms that load most on the second principal component
PCb_plot_idx = np.concatenate((remain_idx[:n_terms], remain_idx[-n_terms:]))

# Combine the indices
PC_plot_idx = np.unique(np.concatenate((PCa_plot_idx, PCb_plot_idx)))

# Get the term names
PC_plot_names = type_names[assoc_idx][PC_plot_idx]

# Get the term loadings
PC_plot_load = L[PC_plot_idx,:]
```

In [42]:

```
### Plot the socio-symbolic constellation

fig, ax = plt.subplots(1,1, figsize = (14,8), constrained_layout=True)

# Title and label text
ax.set_title('PCA of NGO-word matrix ({} NGOs, {} words)'.format(standardized_actor2type_fr
                                                                 standardized_actor2type_fr
                                                                 fontsize = 20,fontweight='
ax.set_xlabel('Principal Component 1 ({:0.0f} % of variation)'.format(100*pca.explained_var
ax.set_ylabel('Principal Component 3 ({:0.0f} % of variation)'.format(100*pca.explained_var

# Grid to mark zero loading
ax.axvline(x=0, c='grey', linestyle='--')
ax.axhline(y=0, c='grey', linestyle='--')

# Plot standardized principal component scores
ax.scatter(scaled_PC[:,0], scaled_PC[:,2], marker = 'v', label='NGO', c='black')

# Plot word loadings
ax.scatter(PC_plot_load[:,0], PC_plot_load[:, 2], marker = 'o', label='Word', c='blue')

# Annotate the plot
texts = []
for x, y, txt in zip(scaled_PC[:,0], scaled_PC[:,2], actor_names):
    texts.append(plt.text(x, y, txt, size=12, weight='bold'))


for x, y, txt in zip(PC_plot_load[:,0], PC_plot_load[:,2], PC_plot_names):
    texts.append(plt.text(x, y, txt, size=12))

adjust_text(texts, arrowprops=dict(arrowstyle="->", color='grey')) # This part is slow

# Set legend to black
plt.legend()
leg = ax.get_legend()
leg.legendHandles[0].set_color('black')
leg.legendHandles[1].set_color('black')

plt.show()
```



In [39]:

```
data_danish.to_csv('data_danish2.csv', index=False)
```

In [40]:

```
data_danish.head()
```

Out[40]:

	actor	tweet	date	retweet	date_convert	@mentions	#hashtags
0	PlanBornefonden	13-årige Larissa bor i Sahel-regionen, og var ...	2021-05-14 09:03:00	NaN	2021-05-14 00:00:00	NaN	NaN
1	PlanBornefonden	Vi ønsker alle muslimer en god Eid i aften! Ei...	2021-05-12 14:00:02	NaN	2021-05-12 00:00:00	NaN	NaN
2	PlanBornefonden	Kom til samtalekøkken med @BosseStine og @Clau...	2021-05-12 11:58:03	@dorthe10	2021-05-12 00:00:00	['BosseStine, ClausMeyerDK']	dkfood
3	PlanBornefonden	Mali, Burkina Faso og Niger - også kendt som d...	2021-05-12 10:00:02	NaN	2021-05-12 00:00:00	NaN	NaN
4	PlanBornefonden	Vores seje kollega Iben Østergaard Markussen f...	2021-05-12 09:23:14	@dorthe10	2021-05-12 00:00:00	['radioloud_dk, MaternityF']	NaN

In [ ]:

In [ ]:

In [ ]:

Appendix 17. LDA

In [2]:

```
#import relevant packages

import pandas as pd
import numpy as np

from datetime import datetime #To check start and end time when running code
from tqdm import tqdm #This is for creating progress bars.
import logging #This is to provide logging of information when running the LDA
import sys #This is to disable logging when it's no longer needed
import pickle #To store and open previously saved machine learning models

#import nlkt libaries
import nltk
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from nltk.stem import LancasterStemmer # Porter is used below. This is an alternative, hars
from nltk.stem import WordNetLemmatizer
from nltk import word_tokenize, pos_tag
from nltk.corpus import wordnet
from nltk.tokenize import TweetTokenizer
import string
from collections import defaultdict

#Importing packages for data visualization
import matplotlib.pyplot as plt
import seaborn as sns
import math
import matplotlib.gridspec as gridspec

#import packages for regular expressions
import regex
import re
```

In [191]:

```
#Importing packages for LDA
from gensim.corpora.dictionary import Dictionary
from gensim.models.ldamulticore import LdaMulticore
from gensim.models.ldamodel import LdaModel
from sklearn.feature_extraction.text import CountVectorizer
from gensim.corpora import Dictionary
from gensim.models.ldamodel import LdaModel
from gensim.models import CoherenceModel
```

In [4]:

```
#import datasets
data_danish = pd.read_csv('data_danish.csv', index_col=0)
```

In [5]:

```
#Duplicates in data

#we choose to drop duplicates whereas we do not risk to make topics based on the same
#tweets and re-tweets.

#dropping duplicates
data_danish_lda = data_danish.drop_duplicates(subset = 'tweet')
```

In [6]:

```
print("Tweets before dropping duplicates:", data_danish.shape)
print("Tweets after dropping duplicates:", data_danish_lda.shape)
```

```
Tweets before dropping duplicates: (9449, 15)
Tweets after dropping duplicates: (9273, 15)
```

In [351]:

```
data_danish_lda.head(3)
```

Out[351]:

	actor	tweet	date	retweet	date_convert	@mentions	#hashtags
0	PlanBornefonden	13-årige Larissa bor i Sahel-regionen, og var ...	2021-05-14 09:03:00	nan	2021-05-14	None	
1	PlanBornefonden	Vi ønsker alle muslimer en god Eid i aften! Ei...	2021-05-12 14:00:02	nan	2021-05-12	None	
2	PlanBornefonden	Kom til samtalekøkken med @BosseStine og @Clau...	2021-05-12 11:58:03	@dorthe10	2021-05-12	[BosseStine, ClausMeyerDK]	dkfood

In [352]:

```
# checking a string with multiple hashtags
data_danish_lda['#hashtags'][9446]
```

Out[352]:

```
' Caritas dkaid humanitarian '
```

In [353]:

```
#removing whitespaces and popular hashtags
string = []
for element in data_danish_lda['#hashtags']:
  item = re.sub(',', ' ', element) # remove ,
  item = re.sub('-19', '', item) # remove 19 from covid-19
  item = re.sub('19', '', item) # remove 19 from covid-19

  item = item.lower() #lower all text
  item = re.sub('dkpol', '', item) #remove hashtag
  item = re.sub('_', '', item)
  item = re.sub('dkaid', '', item) #remove hashtag
  item = re.sub(r'\s+', " ", item) #remove more whitespaces
  item = item.strip() #remove leading and last whitespace
  #item = re.sub('[^a-zA-ZÆæØøÅå]+', ' ', element) # only keeping letters
  string.append(item)


data_danish_lda['hashtags_clean'] = string
```

```
<ipython-input-353-309197346355>:18: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/
stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pand
as.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-v
ersus-a-copy)
  data_danish_lda['hashtags_clean'] = string
```

In [174]:

```
data_danish_lda['hashtags_clean'][9446]
```

Out[174]:

```
'caritas humanitarian'
```

We do not make bigrams of hashtags, as we methodological do not perceive hashtags as words, where grouping words appearing next to each other can be bring meaning and differetn meanings to a word.

In [354]:

```
#Defining NLTK's TweetTokenizer
tokenizer = TweetTokenizer()

tqdm.pandas() #Creates a progress bar. Use progress_apply instead of apply.

#Creating a column of unigrams from the stemmed tweet text
data_danish_lda['hashtags_token'] = data_danish_lda['hashtags_clean'].progress_apply(lambda
```

```
100%|██████████| 9273/9273 [00:00<00:00, 55969.93it/s]
<ipython-input-354-16df89138396>:7: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/
stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pand
as.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-v
ersus-a-copy)
  data_danish_lda['hashtags_token'] = data_danish_lda['hashtags_clean'].prog
ress_apply(lambda x: tokenizer.tokenize(x))
```

In [355]:

```
#Combining unigrams, hashtags and bigrams in one column
data_danish_lda['tokens'] = data_danish_lda.lemmas+data_danish_lda.bigrams+data_danish_lda.
```

```
<ipython-input-355-9dfa68ae85ce>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/
stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pand
as.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-v
ersus-a-copy)
  data_danish_lda['tokens'] = data_danish_lda.lemmas+data_danish_lda.bigrams
+data_danish_lda.hashtags_token
```

In [356]:

```
#after inspecting the topics we can see that the word "barn" occurs in most of them
#therefore, we try removing them
data_danish_lda["tokens"]
words_no_f = []
for lst in data_danish_lda["tokens"]:
    words_no_f.append([word for word in lst if word != "f"])

#If you want "barn" in the LDA, outcomment this snip of code
data_danish_lda["tokens"]
words_no_f = []
for lst in data_danish_lda["tokens"]:
    words_no_f.append([word for word in lst if word != "no"])
```

Notably, it is harder to apply coherent models to a corpus with short documents as we do with tweets (which are very short) than longer documents.

In the LDA model, we set alpha as auto, which mean that the alpha is both low and asymmetric (favourite topics in docs). Notably, we ackknowledge that every doc has a preference --> docs have dominated topics and we let the model distribute the topics by this

We are interested in what the NGOs talk about, eg. the topics. Consequently, we have remove the #'s before the hashtags treating the hashtags as regular words. This is due to the methodological choice that we consider words and hashtags as the same "value" when creating topics.

Preparing the LDA

In [365]:

```
#Create a id2word dictionary

#Insert the column where you saved unigram and bigram tokens between the parentheses
id2word = Dictionary(data_danish_lda['tokens'])

#Viewing how many words are in our vocabulary
print(len(id2word))
#Removing very frequent and infrequent words
id2word.filter_extremes(no_below=10, no_above=.999, keep_n = None)
print(len(id2word))

#Creating a corpus object

corpus = [id2word.doc2bow(doc) for doc in data_danish_lda['tokens']]


# build a dictionary where for each tweet, each word has its own id.
# We have xx tweets
tweets_dictionary = Dictionary(data_danish_lda.tokens)

# Here we build the corpus i.e. vectors with the number of occurence of each word per tweet
tweets_corpus = [tweets_dictionary.doc2bow(tweet) for tweet in data_danish_lda.tokens]
```

```
116377
2838
```

In [358]:

```
print(len(corpus))
print(len(tweets_corpus))
```

```
9273
9273
```

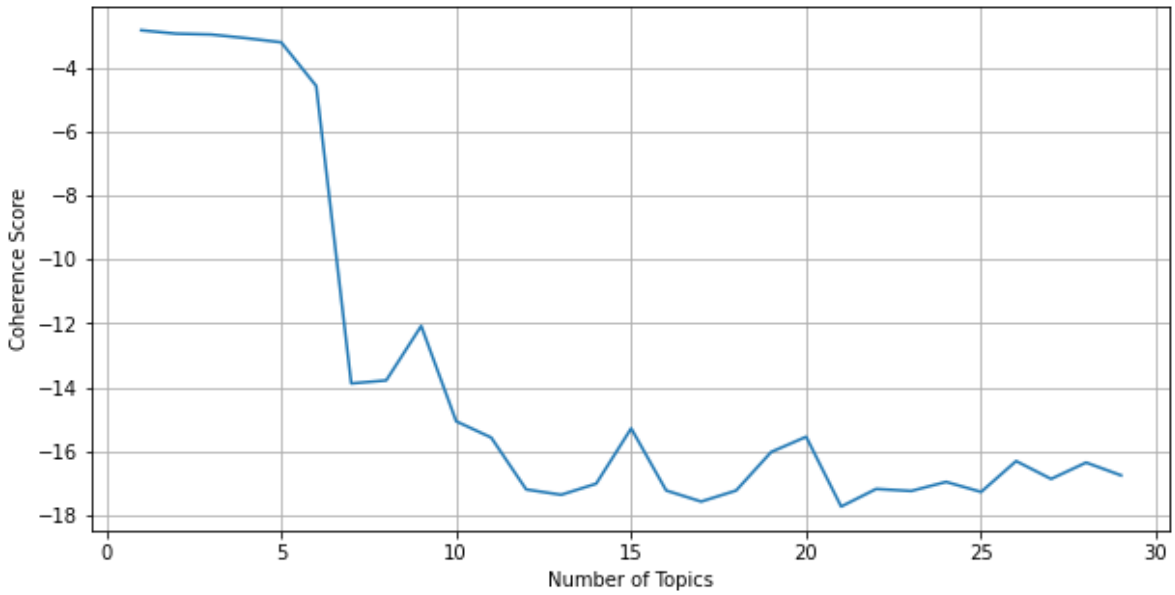
We have explored removing very frequent words by setting no_above to 0.90. However, the results and topics were most distinct beforehand. Since we already have removed stopwords, lemmatized etc., we choose to set no_above to *.999*

In [161]:

```
# Make a for loop creating lda_models with different amount of topics
# compute coherence
tweets_coherence = []

for nb_topics in range(1,30):
    lda = LdaModel(tweets_corpus, num_topics = nb_topics, id2word = tweets_dictionary,
                   passes=10, alpha='auto')
    #^alpha='auto'
    cohm = CoherenceModel(model=lda, corpus=tweets_corpus, dictionary=tweets_dictionary, co
    coh = cohm.get_coherence()
    tweets_coherence.append(coh)
```

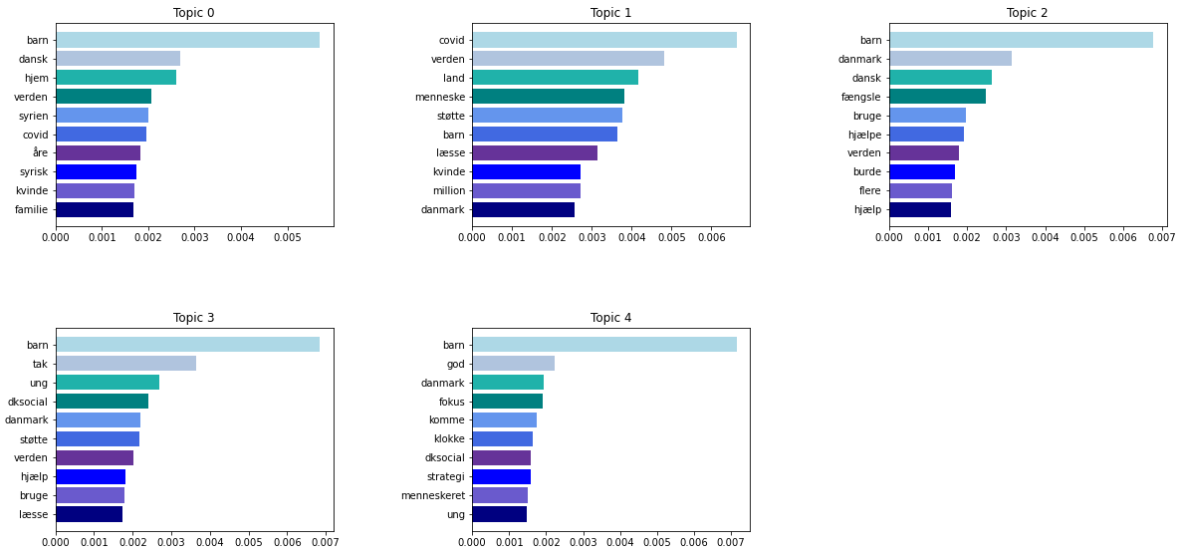
In [162]:

```
# visualize coherence
plt.figure(figsize=(10,5))
plt.plot(range(1,30),tweets_coherence)
plt.xlabel("Number of Topics")
plt.ylabel("Coherence Score")
plt.grid(zorder=3)
#plt.show()

plt.savefig('coherence_score_autoalpha_hashtags_09_test.png')
```



In [368]:

```
# 5 topics
k = 5
tweets_lda = LdaModel(tweets_corpus, num_topics = k, id2word = tweets_dictionary,
                      passes=20, alpha='auto')

#BETA PARAMETERS
def plot_top_words(lda=tweets_lda, nb_topics=k, nb_words=10):
    top_words = [[word for word,_ in lda.show_topic(topic_id, topn=50)] for topic_id in ran
    top_betas = [[beta for _,beta in lda.show_topic(topic_id, topn=50)] for topic_id in ran

    gs  = gridspec.GridSpec(round(math.sqrt(k))+1,round(math.sqrt(k))+1)
    gs.update(wspace=0.5, hspace=0.5)
    #Plot the data:

    plt.figure(figsize=(20,15))
    for i in range(nb_topics):
        ax = plt.subplot(gs[i])
        plt.barh(range(nb_words), top_betas[i][:nb_words], align='center',
                 color=('lightblue', 'lightsteelblue', 'lightseagreen', 'teal', 'cornflower
                        'rebeccapurple','blue', 'slateblue', 'navy'), ecolor='black')
        ax.invert_yaxis()
        ax.set_yticks(range(nb_words))
        ax.set_yticklabels(top_words[i][:nb_words])
        plt.title("Topic "+str(i))
    plt.savefig('til test.png')

plot_top_words(tweets_lda, k, 10)
```



In [369]:

```
#Viewing the shape of base_model.get_topics() to verify that the shape is (topics, tokens)
tweets_lda.get_topics().shape
```

Out[369]:

```
(5, 116377)
```

Investigating the LDA: Gamma parameters

In [372]:

```
data_danish_lda = data_danish_lda.reset_index()
data_danish_lda.tail(3)
```

Out[372]:

	level_0	index	actor	tweet	date	retweet	date_convert	@
9270	9270	9446	CaritasDanmark	Dolores Halpin-Bachmann har arbejdet for #Cari...	2020-02-07 10:49:47	nan	2020-02-07	
9271	9271	9447	CaritasDanmark	Børn skal være børn - https://t.co/h5xNWh7a7U ...	2020-01-31 13:48:55	nan	2020-01-31	
9272	9272	9448	CaritasDanmark	”Ansvaret hviler i høj grad på jeres generatio...	2020-01-21 12:00:47	nan	2020-01-21	

3 rows × 21 columns

In [373]:

```
#Insert the corpus of documents in bag of word format and get a list of (topic, probability
document_topics = list(tweets_lda.get_document_topics(tweets_corpus))
```

In [374]:

```
len(document_topics)
```

Out[374]:

```
9273
```

In [381]:

```
#Creating a list of names for all 10 topics
topics = ['topic_{}'.format(t) for t in range(0,5)]

#Creating a dataframe of gamma probabilities
gamma_probs = pd.DataFrame(np.zeros((len(document_topics),5)), columns = topics)

for i, doc in enumerate(document_topics):
    for pair in doc:
        gamma_probs.loc[i,'topic_{}'.format(pair[0])] = pair[1]
```

In [383]:

```
gamma_probs.shape
```

Out[383]:

```
(9273, 5)
```

In [382]:

```
gamma_probs.tail()
```

Out[382]:

	topic_0	topic_1	topic_2	topic_3	topic_4
9268	0.00000	0.000000	0.000000	0.986477	0.000000
9269	0.00000	0.237831	0.000000	0.757490	0.000000
9270	0.00000	0.988638	0.000000	0.000000	0.000000
9271	0.01155	0.269588	0.010121	0.698640	0.010099
9272	0.00000	0.000000	0.000000	0.880649	0.113925

In [385]:

```
#Merging with the original dataframe
gamma_df = pd.concat((data_danish_lda,gamma_probs), axis = 1)

#Viewing the concatenated dataframe
gamma_df.head()
```

Out[385]:

	level_0	index	actor	tweet	date	retweet	date_convert	@ment
0	0	0	PlanBornefonden	13-årige Larissa bor i Sahel-regionen, og var ...	2021-05-14 09:03:00	nan	2021-05-14	N
1	1	1	PlanBornefonden	Vi ønsker alle muslimer en god Eid i aften! Ei...	2021-05-12 14:00:02	nan	2021-05-12	N
2	2	2	PlanBornefonden	Kom til samtalekøkken med @BosseStine og @Clau...	2021-05-12 11:58:03	@dorthe10	2021-05-12	[BosseS ClausMeye
3	3	3	PlanBornefonden	Mali, Burkina Faso og Niger - også kendt som d...	2021-05-12 10:00:02	nan	2021-05-12	N
4	4	4	PlanBornefonden	Vores seje kollega Iben Østergaard Markussen f...	2021-05-12 09:23:14	@dorthe10	2021-05-12	[radiolouc Materr

5 rows × 26 columns

In [390]:

```
gamma_df.shape
```

Out[390]:

```
(9273, 24)
```

In [391]:

```
gamma_df.to_csv('gamma_df_lda_tweets_5topics.csv')
```

Vizualizing a topic over time

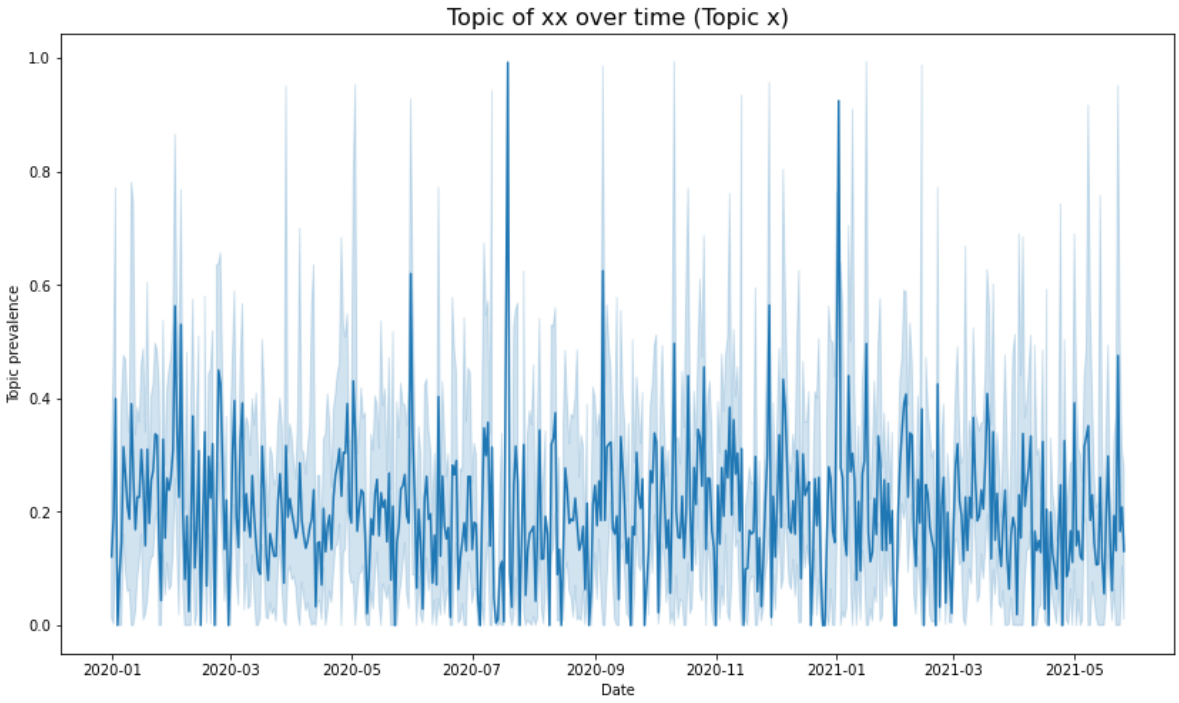
In [392]:

```
#Using regex to find all words and filter them from the weights
words = [re.findall(r'"([^"]*)"',t[1]) for t in tweets_lda.print_topics(50,10)]
for id, t in enumerate(words):
    print(f"------ Topic {id} ------")
    print(' '.join(t), end="\n\n")
```

```
------ Topic 0 ------
barn dansk hjem verden syrien covid åre syrisk kvinde familie

------ Topic 1 ------
covid verden land menneske støtte barn læsse kvinde million danmark

------ Topic 2 ------
barn danmark dansk fængsle bruge hjælpe verden burde flere hjælp

------ Topic 3 ------
barn tak ung dksocial danmark støtte verden hjælp bruge læsse

------ Topic 4 ------
barn god danmark fokus komme klokke dksocial strategi menneskeret ung
```

In [398]:

```
#Setting a theme

#Plotting a figure and setting figure size
plt.figure(figsize = (14,8))

#Choosing colors
palette = ['blue','red']

#Plotting the covid topic for the full dataset
sns.lineplot(x = 'date_convert', y = 'topic_3', data = gamma_df, palette = palette)

plt.title('Topic of xx over time (Topic x)', fontsize = 16)
plt.ylabel('Topic prevalence')
plt.xlabel('Date')

plt.show()
```



LDA - aggregated tweets pr actor

In [406]:

```
data_danish_lda.head(2)
```

Out[406]:

	actor	tweet	date	retweet	date_convert	@mentions	#hashtags	emojis
0	PlanBornefonden	13-årige Larissa bor i Sahel-regionen, og var ...	2021-05-14 09:03:00	nan	2021-05-14	None		
1	PlanBornefonden	Vi ønsker alle muslimer en god Eid i aften! Ei...	2021-05-12 14:00:02	nan	2021-05-12	None		

In [407]:

```
#data_danish_lda['tokens_str'] = data_danish_lda['tokens'].apply(str)
data_danish_lda['tokens_str'] = [' '.join(map(str, l)) for l in data_danish_lda['tokens']]
```

In [408]:

```
tweets_agg = data_danish_lda.groupby(['actor'], as_index = False).agg({'tweet': ' '.join,
                                                                       'tokens_str': ' '.joi
tqdm.pandas() #Creates a progress bar. Use progress_apply instead of apply.

#Creating a column of unigrams from the stemmed tweet text
tweets_agg['tokens'] = tweets_agg['tokens_str'].progress_apply(lambda x: tokenizer.tokenize
```

```
100%|██████████| 27/27 [00:01<00:00, 13.86it/s]
```

In [409]:

```
#removing whitespaces and popular hashtags
string = []
for element in tweets_agg['tokens_str']:
  item = re.sub('\+', ' ', element) # remove ,
  item = re.sub(r'\s+', " ", item) #remove more whitespaces

  string.append(item)


tweets_agg['tokens_str'] = string
```

In [410]:

```
#Creating a column of unigrams from the stemmed tweet text
tweets_agg['tokens'] = tweets_agg['tokens_str'].progress_apply(lambda x: tokenizer.tokenize
```

```
100%|██████████| 27/27 [00:01<00:00, 13.80it/s]
```

In [411]:

```
#after inspecting the topics we can see that the word "barn" occurs in most of them
#therefore, we try removing them

words_no_s = []
for lst in tweets_agg['tokens']:
    words_no_s.append([word for word in lst if word != "s"])

tweets_agg['tokens'] = words_no_s


words_no_ = []
for lst in tweets_agg['tokens']:
    words_no_.append([word for word in lst if word != "+"])
tweets_agg['tokens'] = words_no_

words_no_1 = []
for lst in tweets_agg['tokens']:
    words_no_1.append([word for word in lst if word != " "])
tweets_agg['tokens'] = words_no_1

words_no_1 = []
for lst in tweets_agg['tokens']:
    words_no_1.append([word for word in lst if word != "|"])
tweets_agg['tokens'] = words_no_1

words_no_1 = []
for lst in tweets_agg['tokens']:
    words_no_1.append([word for word in lst if word != "-"])
tweets_agg['tokens'] = words_no_1
```

In [412]:

```
tweets_agg.head(2)
```

Out[412]:

	actor	tweet	tokens_str	tokens
0	ADRA_Danmark	Humanitær støtte til #Syrien sker i tæt samarb...	humanitær støtte ske samarbejde gode partner t...	[humanitær, støtte, ske, samarbejde, gode, par...
1	AVestegnen	Tænk at dine børn hopper på trampolin, mens de...	tænke din barn hoppe trampolin men ske bombeek...	[tænke, din, barn, hoppe, trampolin, men, ske,...

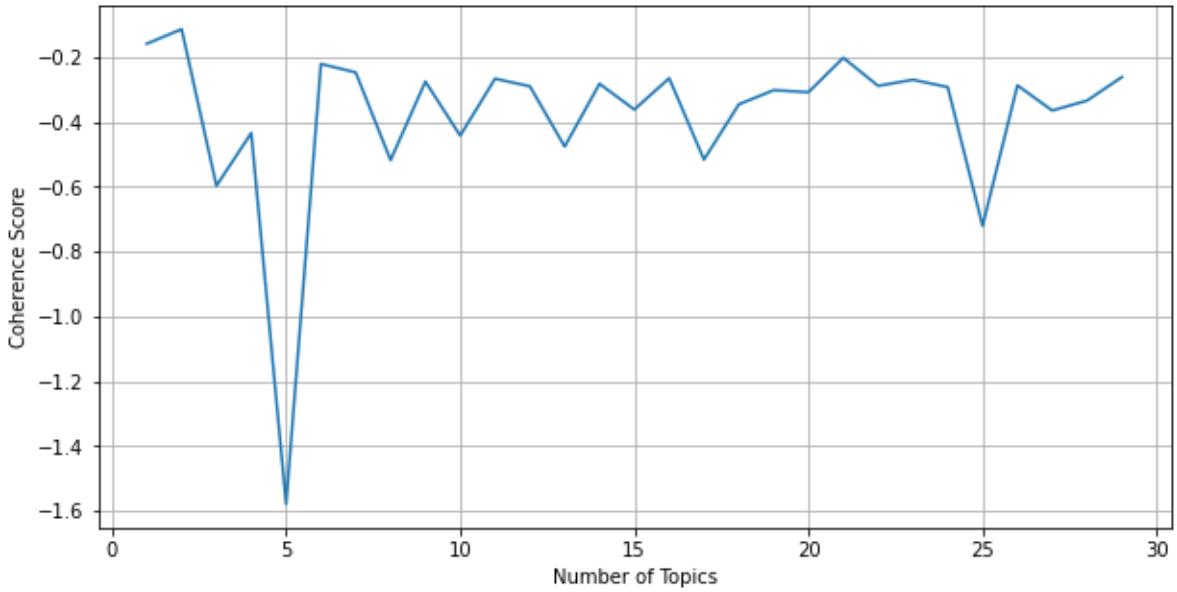
In [413]:

```
#Create a id2word dictionary

#Insert the column where you saved unigram and bigram tokens between the parentheses
id2word = Dictionary(tweets_agg['tokens'])

#Viewing how many words are in our vocabulary
print('Length of id2word:', len(id2word))
#Removing very frequent and infrequent words
id2word.filter_extremes(no_below=10, no_above=.999, keep_n = None)
print('Length of id2word after filtering:', len(id2word))

#Creating a corpus object

corpus = [id2word.doc2bow(doc) for doc in tweets_agg['tokens']]


# build a dictionary where for each tweet, each word has its own id.
# We have xx tweets
tweets_dictionary = Dictionary(tweets_agg.tokens)

# Here we build the corpus i.e. vectors with the number of occurence of each word per tweet
tweets_corpus = [tweets_dictionary.doc2bow(tweet) for tweet in tweets_agg.tokens]

print('Length of corpus:',len(corpus))
print(len(tweets_corpus))
```

```
Length of id2word: 116060
Length of id2word after filtering: 1273
Length of corpus: 27
27
```

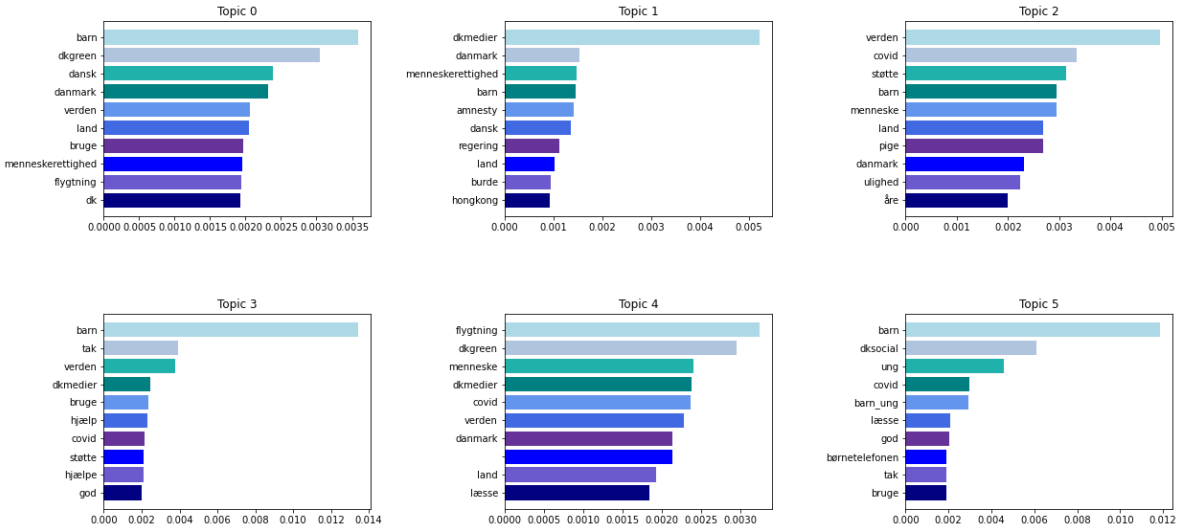
In [340]:

```
# Make a for loop creating lda_models with different amount of topics
# compute coherence
tweets_coherence = []

for nb_topics in range(1,30):
    lda = LdaModel(tweets_corpus, num_topics = nb_topics, id2word = tweets_dictionary,
                   passes=10, alpha='auto')
    #^alpha='auto'
    cohm = CoherenceModel(model=lda, corpus=tweets_corpus, dictionary=tweets_dictionary, co
    coh = cohm.get_coherence()
    tweets_coherence.append(coh)

# visualize coherence
plt.figure(figsize=(10,5))
plt.plot(range(1,30),tweets_coherence)
plt.xlabel("Number of Topics")
plt.ylabel("Coherence Score")
plt.grid(zorder=3)
#plt.show()

plt.savefig('coherence_score_autoalpha_aggregated.png')
```



In [415]:

```
#10 topics with hashtags
k = 6
tweets_lda = LdaModel(tweets_corpus, num_topics = k, id2word = tweets_dictionary,
                      passes=20, alpha='auto')

#BETA PARAMETERS
def plot_top_words(lda=tweets_lda, nb_topics=k, nb_words=10):
    top_words = [[word for word,_ in lda.show_topic(topic_id, topn=50)] for topic_id in ran
    top_betas = [[beta for _,beta in lda.show_topic(topic_id, topn=50)] for topic_id in ran

    gs  = gridspec.GridSpec(round(math.sqrt(k))+1,round(math.sqrt(k))+1)
    gs.update(wspace=0.5, hspace=0.5)
    #Plot the data:

    plt.figure(figsize=(20,15))
    for i in range(nb_topics):
        ax = plt.subplot(gs[i])
        plt.barh(range(nb_words), top_betas[i][:nb_words], align='center',
                 color=('lightblue', 'lightsteelblue', 'lightseagreen', 'teal', 'cornflower
                        'rebeccapurple','blue', 'slateblue', 'navy'), ecolor='black')
        ax.invert_yaxis()
        ax.set_yticks(range(nb_words))
        ax.set_yticklabels(top_words[i][:nb_words])
        plt.title("Topic "+str(i))
    plt.savefig('6_topics_auto_agg.png')

plot_top_words(tweets_lda, k, 10)
```



Gamma parameters for model on actor level

In [416]:

```
tweets_agg = tweets_agg.reset_index()
tweets_agg.tail(3)
```

Out[416]:

	index	actor	tweet	tokens_str	tokens
24	24	noedhjaelp	Tak for den store omsorg og opbakning ifm den ...	tak stor omsorg opbakning ifm personlig traged...	[tak, stor, omsorg, opbakning, ifm, personlig,...
25	25	oxfamibis	Vi er dybt bekymrede over situationen i Jerusa...	dybt bekymre situation jerusalem far yderliger...	[dybt, bekymre, situation, jerusalem, far, yde...
26	26	redbarnetdk	Børnene er altid de mest udsatte – også i konf...	barn altid udsat konflikt barn angst barrikade...	[barn, altid, udsat, konflikt, barn, angst, ba...

In [417]:

```
#Insert the corpus of documents in bag of word format and get a list of (topic, probability
document_topics = list(tweets_lda.get_document_topics(tweets_corpus))
len(document_topics)
#Creating a list of names for all 10 topics
topics = ['topic_{}'.format(t) for t in range(0,6)]

#Creating a dataframe of gamma probabilities
gamma_probs = pd.DataFrame(np.zeros((len(document_topics),6)), columns = topics)

for i, doc in enumerate(document_topics):
    for pair in doc:
        gamma_probs.loc[i,'topic_{}'.format(pair[0])] = pair[1]

gamma_probs.tail()
#gamma_df = gamma_df.drop(columns=['level_0', 'index'])
#Merging with the original dataframe
gamma_df = pd.concat((tweets_agg,gamma_probs), axis = 1)

#Viewing the concatenated dataframe
gamma_df.head()
gamma_df.shape

gamma_df.to_csv('gamma_df_lda_actor_aggtweets_6topics.csv')
```

Appendix 18. hSBM

TopSBM: Topic Modeling with Stochastic Block Models - made in colab (do NOT try to run in Jupyter notebook)

In [1]:

```
# install dependencies
!pip install -q condacolab
import condacolab
condacolab.install()

import condacolab
#condacolab.check()

! conda config --add channels conda-forge
! conda config --add channels ostrokach-forge
! conda config --add channels pkgw-forge

! conda install gtk3
! conda install pygobject graph-tool cairo
! conda install -c conda-forge graph-tool
! git clone https://github.com/martingerlach/hSBM_Topicmodel.git
```

```
✨🏰✨ Everything looks OK!
Warning: 'conda-forge' already in 'channels' list, moving to the top
Warning: 'ostrokach-forge' already in 'channels' list, moving to the top
Warning: 'pkgw-forge' already in 'channels' list, moving to the top
Collecting package metadata (current_repodata.json): done
Solving environment: done

# All requested packages already installed.

Collecting package metadata (current_repodata.json): done
Solving environment: done

# All requested packages already installed.

Collecting package metadata (current_repodata.json): done
Solving environment: done

# All requested packages already installed.

fatal: destination path 'hSBM_Topicmodel' already exists and is not an empty
directory.
```

In [2]:

```
%load_ext autoreload
%autoreload 2

import os
import pylab as plt
%matplotlib inline
import graph_tool.all as gt
from hSBM_Topicmodel.sbmtm import sbmtm
```

In [3]:

```
# load dataset - loading the aggregated dataset that has been through the process up untill
import pandas as pd
path2tox_data = '/content/gdrive/My Drive/Deep learning Spring 2021/data_danish2.csv' # cha
twitter_data2 = pd.read_csv(path2tox_data)
```

In [10]:

```
print(type(twitter_data2['proc_freq'][0]))
print(type(twitter_data2['proc_text_all'][0]))
```

```
<class 'str'>
<class 'str'>
```

In [5]:

```
#Make column in to string
twitter_data2['proc_freq'] = twitter_data2['proc_freq'].apply(str)
```

In [69]:

```
twitter_data2
```

Out[69]:

	actor	tweet	date	retweet	date_convert	@mentions
0	PlanBornefonden	13-årige Larissa bor i Sahel-regionen, og var ...	2021-05-14 09:03:00	NaN	2021-05-14 00:00:00	NaN
1	PlanBornefonden	Vi ønsker alle muslimer en god Eid i aften! Ei...	2021-05-12 14:00:02	NaN	2021-05-12 00:00:00	NaN
2	PlanBornefonden	Kom til samtalekøkken med @BosseStine og @Clau...	2021-05-12 11:58:03	@dorthe10	2021-05-12 00:00:00	['BosseStine ClausMeyerDK'
3	PlanBornefonden	Mali, Burkina Faso og Niger - også kendt som d...	2021-05-12 10:00:02	NaN	2021-05-12 00:00:00	NaN
4	PlanBornefonden	Vores seje kollega Iben Østergaard Markussen f...	2021-05-12 09:23:14	@dorthe10	2021-05-12 00:00:00	['radioloud_dk MaternityF'
...	...	...	...	...	...	..
9444	CaritasDanmark	Vores Internationale chef Betina Gollander-Jen...	2020-03-03 13:46:14	NaN	2020-03-03 00:00:00	NaN
9445	CaritasDanmark	#askeonsdag \nFasteindsamling https://t.co/...	2020-02-26 12:26:26	NaN	2020-02-26 00:00:00	['Pesitho1'
9446	CaritasDanmark	Dolores Halpin-Bachmann har arbejdet for #Cari...	2020-02-07 10:49:47	NaN	2020-02-07 00:00:00	NaN
9447	CaritasDanmark	Børn skal være børn - https://t.co/h5xNWh7a7U ...	2020-01-31 13:48:55	NaN	2020-01-31 00:00:00	NaN
9448	CaritasDanmark	”Ansvaret hviler i høj grad på jeres generatio...	2020-01-21 12:00:47	NaN	2020-01-21 00:00:00	NaN

9449 rows × 18 columns

In [28]:

```
#groupby actor and aggregate on proc_freq column to only run model on agregated data
twitter_agg = twitter_data2.groupby(['actor'], as_index = False).agg({'proc_freq':' '.join}
twitter_agg
```

Out[28]:

	actor	proc_freq
0	ADRA_Danmark	humanitær støtte ske samarbejde gode partner t...
1	AVestegnen	tænke din barn men ske side by din_barn vide ...
2	ActionAidDK	endelig blive tid igen glæde gade grønt fælles...
3	BornsVilkar	gode tiltag udspille savne fokus forebyggelse ...
4	CARE_Danmark	fejre år fødselsdag care skabe hjælpe sulte eu...
5	CaritasDanmark	bruge magte hør generalsekretær tale weekend k...
6	DKIndsamling	klokke hej igen forkert fn unhcr styre tal fi...
7	DRC_dk	forhindre europæisk myndighed ulovlig menneske...
8	DignityDK	dansk såkaldt betaler dermed middelhavet sende...
9	MissionEast	nepal indien fn kalde akut indsats coronapande...
10	PlanBornefonden	årig bor sahelregion tvinge flygte landsby man...
11	RTC_DK	godt verden kompleks stadig barn mor syrisk ho...
12	RefWelcome	syrer uafhængig flygtningenævn brug læsse rapp...
13	UNDP_Danmark	vigtig nogensinde stå sammen person bekæmpe ur...
14	UNICEFDK	koordinere social mediere samtidig god din med...
15	WFP_DK	million menneske sulten seng men hver åre spil...
16	amnestydk	israel angribe undersøge dokumentere angribe c...
17	blaakorsdanmark	flere kæmpe misbruge psykisk mistrivsel ensomh...
18	danmissiondk	denmark invitere myanmar maj klokke kbh myanma...
19	danskrodekors	frivillig social engagement dk stort få frivil...
20	globaltfokus	bæredygtig global forandring udviklingssamarbe...
21	lgbt_asylum	ses demo slå ringe syrer danmark eid medlem v...
22	menneskeret	bekæmpe corona uden privatliv retssikkerhed gi...
23	msf_dk	ilte dø kritisk syg patient oplever hospital k...
24	noedhjaelp	tak stor omsorg opbakning ifm personlig traged...
25	oxfamibis	dybt bekymre situation far yderligere familie ...
26	redbarnetdk	barn altid udsat konflikt barn angst bange for...

In [29]:

```
#Helper function for preparing documents
#prepare documents
import re
url_re = re.compile('(?:https?:://)?www.[^ ]+')
import nltk
tokenizer = nltk.tokenize.casual.TweetTokenizer()

def preprocess_doc(string,tokenizer=tokenizer.tokenize):
    "simple preprocessing function"
    doc = string.lower() # lowercase
    doc = url_re.sub('url',doc)
    doc = tokenizer(doc)
    return doc

twitter_agg['doc'] = twitter_agg.proc_freq.apply(preprocess_doc).values
docs2 = twitter_agg['doc']
# remove infrequent words.
from collections import Counter
cutoff = 10
c = Counter()
for doc in docs2:
    c.update(Counter(doc))
vocab = c.most_common(25000)
vocab = set([w for w,count in vocab if count>5])
# remove words
docs2 = [[w for w in doc if w in vocab] for doc in docs2]
```

In [30]:

```
## we create an instance of the sbmtm-class
mode2 = sbmtm()
## we have to create the word-document network from the corpus
mode2.make_graph(docs2,documents=['%d'%i for i in range(len(docs2))])
```

In [92]:

```
mode2
```

Out[92]:

```
<hSBM_Topicmodel.sbmtm.sbmtm at 0x7ff169bf1d90>
```

In [31]:

```
## fit the model
gt.seed_rng(32) ## seed for graph-tool's random number generator --> same results
mode2.fit()
```

In [32]:

```
mode2.topics()
```

Out[32]:

```
{0: [('humanitær', 0.4289655172413793),
  ('konflikt', 0.22206896551724137),
  ('made', 0.2),
  ('humanitær_kris', 0.057931034482758624),
  ('uddele', 0.057931034482758624),
  ('spredning', 0.03310344827586207)],
 1: [('støtte', 0.6919642857142857),
  ('hver', 0.20625),
  ('muligt', 0.10178571428571428)],
 2: [('burde', 0.23227247347850363),
  ('ansvar', 0.0993858179787828),
  ('ske', 0.07091010608598548),
  ('desværre', 0.06867671691792294),
  ('forholde', 0.06420993858179788),
  ('ja', 0.04913456169737577),
  ('europa', 0.047459519821328865),
  ('forslag', 0.04690117252931323),
```

In [33]:

```
#Dictionary of topics
model2_dict = mode2.topics()
```

In [130]:

```
#Making a dictionary of top 5 topics related to each actor
topics_dict1 = {}
list_of_actors = list(twitter_agg.actor.unique())

for i in range(len(list_of_actors)):
  i_doc = i
  topic = mode2.topicdist(i_doc)
  topic = dict(topic)
  topic = sorted(topic.items(), key=lambda x: x[1], reverse=True)
  topic = topic[:5]
  topics_dict[list_of_actors[i]] = topic
```

In [131]:

```
#Taking the unique topics to see which topics the NGOs use
unique_topics = []
for y,k in topics_dict.items():
  tops = []
  for i,j in k:
    print(i)
    unique_topics.append(i)

  #unique_topics = tops

unique = pd.DataFrame(unique_topics,columns=['Values'])
unique_topics = unique.Values.unique()
```

```
0
6
20
3
4
32
54
67
53
10
8
32
60
82
153
125
41
20
18
```

In [153]:

```
#Sorting unique topics
sorted(unique_topics)
```

Out[153]:

```
[0,
 3,
 4,
 6,
 7,
 8,
 10,
 12,
 14,
 15,
 18,
 19,
 20,
 25,
 26,
 28,
 29,
 32,
 36,
 40,
 41,
 46,
 49,
 53,
 54,
 60,
 67,
 74,
 75,
 76,
 77,
 80,
 82,
 83,
 94,
 97,
 98,
 99,
 101,
 105,
 110,
 113,
 116,
 121,
 123,
 125,
 127,
 130,
 131,
 134,
 136,
 137,
 145,
```

```
 148,
 153,
 159,
 162,
 165,
 168,
 170,
 172,
 175]
```

In [82]:

```
type(topics_dict['ADRA_Danmark'][0][0])
topics_dict['ADRA_Danmark'][0][0]
```

Out[82]:

```
0
```

In [134]:

```
#Making dataframe of topics dictionary
df3 = pd.DataFrame(topics_dict)
df3
```

Out[134]:

	ADRA_Danmark	AVestegnen	ActionAidDK	BornsVilkar
0	(0, 0.04918032786885246)	(32, 0.04361873990306947)	(8, 0.044670717687633435)	(125, 0.08345738742091552)
1	(6, 0.04918032786885246)	(54, 0.04254173397953689)	(32, 0.04122187551322056)	(41, 0.06122069222180871)
2	(20, 0.04918032786885246)	(67, 0.04092622509423802)	(60, 0.030054196091312203)	(20, 0.042705619650167476)
3	(3, 0.03278688524590164)	(53, 0.03446418955304254)	(82, 0.02874035145344063)	(18, 0.03917007815407518)
4	(4, 0.03278688524590164)	(10, 0.029617662897145933)	(153, 0.028411890293972737)	(131, 0.03275027912169706)

In [141]:

```
#Saving dataframe as data_topics.csv
df3.to_csv('/content/gdrive/My Drive/Deep learning Spring 2021/data_topics.csv')
```

In [36]:

```
#Inscepting clusters amongst the NGOs (documents)
mode2.clusters()
```

Out[36]:

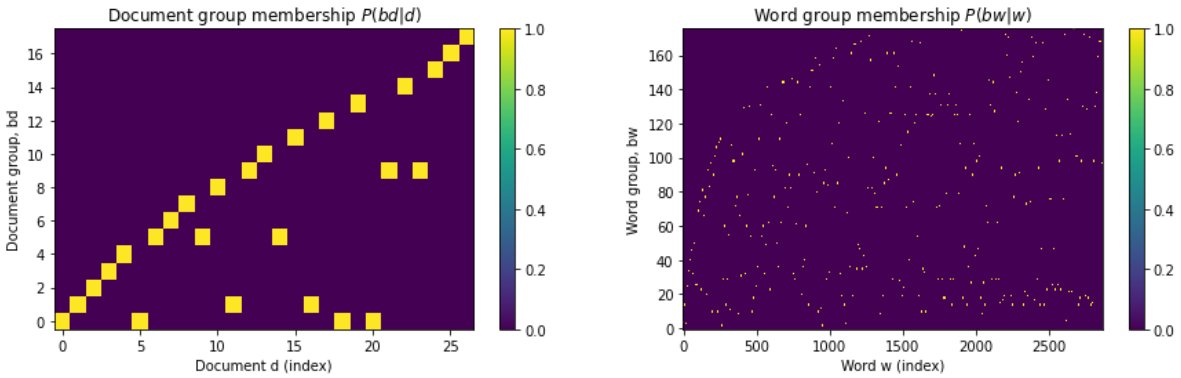
```
{0: [('0', 1.0), ('20', 1.0), ('5', 1.0), ('18', 1.0)],
 1: [('1', 1.0), ('16', 1.0), ('11', 1.0)],
 2: [('2', 1.0)],
 3: [('3', 1.0)],
 4: [('4', 1.0)],
 5: [('6', 1.0), ('9', 1.0), ('14', 1.0)],
 6: [('7', 1.0)],
 7: [('8', 1.0)],
 8: [('10', 1.0)],
 9: [('23', 1.0), ('21', 1.0), ('12', 1.0)],
 10: [('13', 1.0)],
 11: [('15', 1.0)],
 12: [('17', 1.0)],
 13: [('19', 1.0)],
 14: [('22', 1.0)],
 15: [('24', 1.0)],
 16: [('25', 1.0)],
 17: [('26', 1.0)]}
```

In [157]:

```
#Plotting thee document group memberships
p_td_d,p_tw_w = mode2.group_membership()

plt.figure(figsize=(15,4))
plt.subplot(121)
plt.imshow(p_td_d,origin='lower',aspect='auto',interpolation='none')
plt.title(r'Document group membership $P(bd | d)$')
plt.xlabel('Document d (index)')
plt.ylabel('Document group, bd')
plt.colorbar()

plt.subplot(122)
plt.imshow(p_tw_w,origin='lower',aspect='auto',interpolation='none')
plt.title(r'Word group membership $P(bw | w)$')
plt.xlabel('Word w (index)')
plt.ylabel('Word group, bw')
plt.colorbar()

from google.colab import files
plt.savefig("actor_plots2.png")
files.download("actor_plots2.png")
```

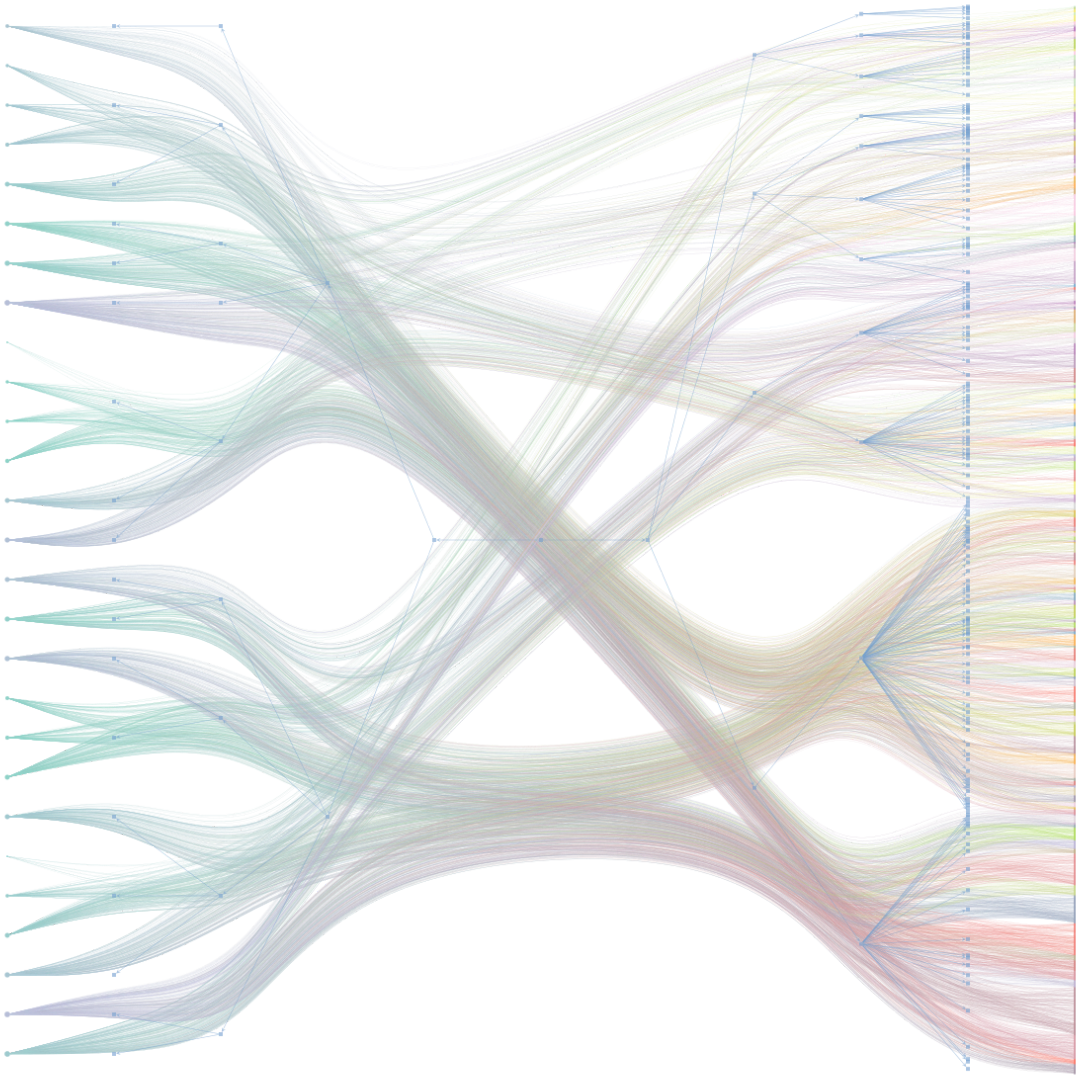
<IPython.core.display.Javascript object>

<IPython.core.display.Javascript object>



In [165]:

```
#Plotting the models topic distribution
mode2.plot(nedges=5000)
```



```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.Javascript object>
```

```
<Figure size 432x288 with 0 Axes>
```

Dictionary

In [28]:

```
#Import libraries
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import re
```

In [31]:

```
#Qualitatively adding names to the top 5 topics for each NGO (only 63 of the topics)
topics_names = {0: "Humanitær krise", 3: "Samarbejde", 4:'Hjælp til udsatte', 6: 'Tak', 7:'
                12:'Vigtigt', 14:'Områder skal modtage', 15:'Fødevarer sikkerhed', 18:'Ople
                20:'Bruge god tid', 25:'Styrke samfundet', 26:'Menneskerettighed', 28:'Udvi
                29:'Positiv', 32:'Stå sammen', 36:'Glad dansker', 40:'Donere til coronaviru
                49:'Lige store', 53:'Forbrydelse', 54:'Demonstration', 60:'Regeringen', 67:
                74:'Asylansøgere og flygtninge', 75:'Danmark', 76:'Tortur', 77:'Risikere',
                83:'Hjem fra Syrien', 94:'Humant lovforslag', 97:'Johanne Schmidt-Nielsen',
                99:'Danske børn i fangelejrer', 101:'Unge pigers rettigheder', 105:'DKmedie
                116:'Vaccine', 121:'Klima', 123:'Syrisk flygtning', 125:'Udsatte børn og un
                131:'Børns rettigheder', 134: 'Hospitals hjælp', 136:'DKsocial', 137:'Kommu
                148:'Alkohol', 153:'DKgreen', 159:'Humanitær politik', 162:'Fattig', 165:'K
                168:'Oxfam Ibis - Skattely', 170:'Hungersnød', 172:'Verdensmålene', 175:'UN
```

In [32]:

```
len(topics_names)
```

Out[32]:

```
62
```

In [33]:

```
#Read in and transpose the dataset containing the aggregated topics
data = pd.read_csv('data_topics.csv', index_col=0)
data = data.transpose()
```

In [34]:

```
data.head()
```

Out[34]:

	0	1	2	
ADRA_Danmark	(0, 0.04918032786885246)	(6, 0.04918032786885246)	(20, 0.04918032786885246)	0.03278
AVestegnen	(32, 0.04361873990306947)	(54, 0.04254173397953689)	(67, 0.04092622509423802)	0.03446
ActionAidDK	(8, 0.044670717687633435)	(32, 0.04122187551322056)	(60, 0.030054196091312203)	0.02874
BornsVilkar	(125, 0.08345738742091552)	(41, 0.06122069222180871)	(20, 0.042705619650167476)	0.03917
CARE_Danmark	(121, 0.055756324212700055)	(153, 0.044269488900361385)	(8, 0.035622096024780586)	0.03265

In [20]:

```
#Function to make string to tuple
def str_to_tup(str):
  m = re.match(r"\s*\(\s*(\d+)\s*,\s*(0\.\d+)\s*\)\s*", str)
  return (int(m.group(1)), float(m.group(2)))
```

In [21]:

```
#Applying function
data[0] = data[0].apply(str_to_tup)
data[1] = data[1].apply(str_to_tup)
data[2] = data[2].apply(str_to_tup)
data[3] = data[3].apply(str_to_tup)
data[4] = data[4].apply(str_to_tup)
```

In [22]:

```
#Adding topic names
data[0] = data[0].apply(lambda x: (topic_names[x[0]], x[1]))
data[1] = data[1].apply(lambda x: (topic_names[x[0]], x[1]))
data[2] = data[2].apply(lambda x: (topic_names[x[0]], x[1]))
data[3] = data[3].apply(lambda x: (topic_names[x[0]], x[1]))
data[4] = data[4].apply(lambda x: (topic_names[x[0]], x[1]))
```

In [23]:

```
#Transpose data again
data = data.transpose()

#tops = data.to_dict()
```

In [24]:

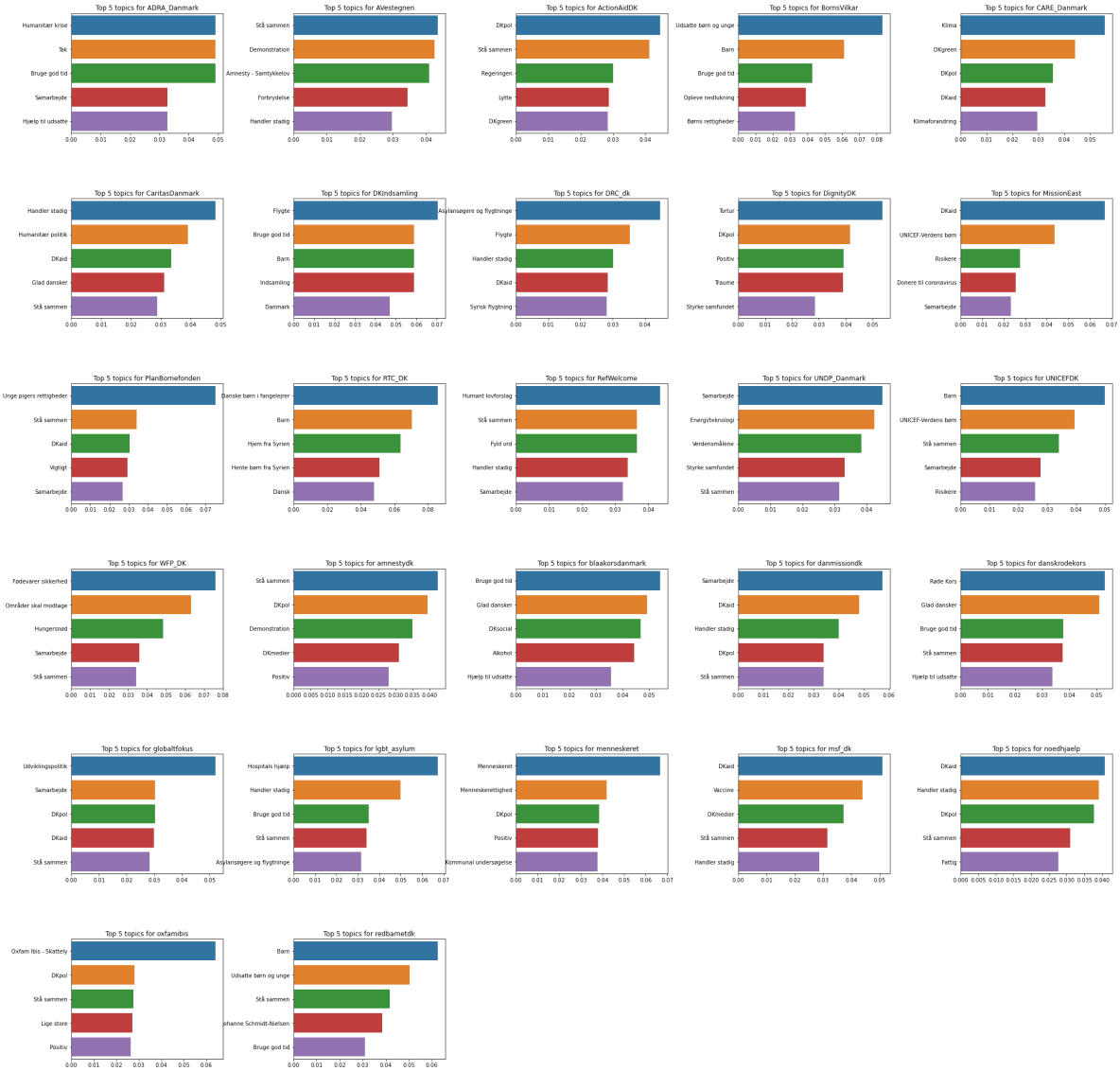
```
#Checking data for one NGO
data["ADRA_Danmark"]
```

Out[24]:

```
0      (Humanitær krise, 0.04918032786885246)
1                  (Tak, 0.04918032786885246)
2        (Bruge god tid, 0.04918032786885246)
3           (Samarbejde, 0.03278688524590164)
4    (Hjælp til udsatte, 0.03278688524590164)
Name: ADRA_Danmark, dtype: object
```

In [25]:

```
#Plotting top 5 topics for each NGO
fig, axes = plt.subplots(nrows = 6, ncols = 5, figsize=(30, 30))
fig.tight_layout(pad=10.0)

for ax, actor in zip(axes.flatten(), data):

    topics = [t[0] for t in data[actor]]
    scores = [t[1] for t in data[actor]]
    sns.barplot(ax = ax, y = topics, x = scores, orient = "h")

    ax.set(title=f'Top 5 topics for {actor}')

axes[5][2].set_visible(False)
axes[5][3].set_visible(False)
axes[5][4].set_visible(False)

plt.savefig('hsbm_topics.png', dpi=600)
```



In [27]:

```
#Plotting words for each topic (and plotting the topics)
fig, axes = plt.subplots(nrows = 18, ncols = 10, figsize=(30, 60),squeeze=False)
fig.tight_layout(pad=10.0)

for ax, (topic, word) in zip(axes.flatten(), dict_topics.items()):
    x = []
    y = []
    for top_num in word:
        y.append(top_num[0])
        x.append(top_num[1])
        #print(top_num[])
    #print(x, y)
    sns.barplot(ax = ax, y = y, x = x, orient='h')

    ax.set(title=f'Words for topic_{topic}')

plt.savefig('topics_tweets_hsbm.png')

        #sns.barplot(x, y, alpha=0.9,ax = axs[r][c])
```

Dict for topics for tweets

In [15]:

```
#Making dictionary for the 176 topics
dict_topics = {0: [('humanitær', 0.4289655172413793),
  ('konflikt', 0.22206896551724137),
  ('made', 0.2),
  ('humanitær_kris', 0.057931034482758624),
  ('uddele', 0.057931034482758624),
  ('spredning', 0.03310344827586207)],
 1: [('støtte', 0.6919642857142857),
  ('hver', 0.20625),
  ('muligt', 0.10178571428571428)],
 2: [('burde', 0.23227247347850363),
  ('ansvar', 0.0993858179787828),
  ('ske', 0.07091010608598548),
  ('desværre', 0.06867671691792294),
  ('forholde', 0.06420993858179788),
  ('ja', 0.04913456169737577),
  ('europa', 0.047459519821328865),
  ('forslag', 0.04690117252931323),
  ('stat', 0.044109436069235064),
  ('krav', 0.03238414293690676)],
 3: [('læsse', 0.24259681093394078),
  ('arbejde', 0.190584662110858),
  ('samarbejde', 0.11845102505694761),
  ('skab', 0.06036446469248292),
  ('uge', 0.05239179954441914),
  ('udfordring', 0.050493545937737284),
  ('gennem', 0.04441913439635535),
  ('hvilken', 0.04100227790432802),
  ('søger', 0.030372057706909643),
  ('hele_verden', 0.025436598329536826)],
 4: [('hjælp', 0.2857142857142857),
  ('udsat', 0.25486250838363517),
  ('gode', 0.19584171696847752),
  ('klare', 0.18376928236083165),
  ('bruge_hjælp', 0.043594902749832326),
  ('kontakte', 0.03621730382293763)],
 5: [('pandemi', 0.2560646900269542),
  ('partner', 0.20754716981132076),
  ('frede', 0.15498652291105122),
  ('program', 0.0876010781671159),
  ('forhindre', 0.06334231805929919),
  ('lever', 0.04582210242587601),
  ('sammenhæng', 0.03638814016172507),
  ('strategisk', 0.03369272237196765),
  ('niveau', 0.03369272237196765),
  ('sidste_uge', 0.03099730458221024)],
 6: [('tak', 0.7008695652173913),
  ('covid', 0.2252173913043478),
  ('fulde', 0.07391304347826087)],
 7: [('dkaid', 1.0)],
 8: [('dkpol', 1.0)],
 9: [('rigtig', 0.229957805907173),
  ('tænke', 0.20253164556962025),
  ('aftale', 0.18354430379746836),
  ('dejlig', 0.10759493670886076),
  ('parti', 0.08227848101265822),
  ('forhandling', 0.05485232067510549),
  ('knappe', 0.05485232067510549),
  ('øvrig', 0.03164556962025317),
```

```
 ('sommer', 0.029535864978902954),
 ('ansøgning', 0.023206751054852322)],
10: [('holde', 0.0759493670886076),
 ('stadig', 0.06715893108298171),
 ('samle', 0.048171589310829814),
 ('organisation', 0.04149085794655415),
 ('nyhed', 0.03762306610407876),
 ('handler', 0.03762306610407876),
 ('vis', 0.02988748241912799),
 ('måde', 0.029535864978902954),
 ('trodse', 0.026371308016877638),
 ('handle', 0.026371308016877638)],
11: [('dag', 0.11550268610897928),
 ('tale', 0.11396776669224866),
 ('hel', 0.11243284727551804),
 ('stærkt', 0.06178050652340752),
 ('presse', 0.0602455871066769),
 ('enig', 0.0602455871066769),
 ('langt', 0.05065234075211052),
 ('huske', 0.047582501918649274),
 ('mangel', 0.04029163468917882),
 ('virkelig', 0.035303146584804296)],
12: [('vigtigt', 0.35768261964735515),
 ('volde', 0.14021830394626364),
 ('særligt', 0.12510495382031905),
 ('ofte', 0.06717044500419815),
 ('en', 0.0654911838790932),
 ('prioritere', 0.05289672544080604),
 ('krop', 0.038623005877413935),
 ('forebyggelse', 0.031066330814441646),
 ('vigtigt_fokus', 0.02434928631402183),
 ('dreng', 0.022670025188916875)],
13: [('kamp', 0.48514851485148514),
 ('men', 0.22607260726072606),
 ('fortsætter', 0.1485148514851485),
 ('jobbe', 0.08085808580858085),
 ('side', 0.0594059405940594)],
14: [('område', 0.0625),
 ('modtage', 0.0586283185840708),
 ('måned', 0.051991150442477874),
 ('yde', 0.03816371681415929),
 ('vide', 0.03263274336283186),
 ('rundt', 0.030973451327433628),
 ('takket', 0.02765486725663717),
 ('advare', 0.024336283185840708),
 ('udgøre', 0.023230088495575223),
 ('forskellig', 0.02267699115044248)],
15: [('wfp', 0.12176165803108809),
 ('fødevare', 0.10103626943005181),
 ('fødevareassistance', 0.07253886010362694),
 ('wfpostscriptum', 0.06347150259067358),
 ('fødevaresikkerhed', 0.06217616580310881),
 ('skolemåltid', 0.04922279792746114),
 ('måltid', 0.04922279792746114),
 ('nære', 0.03626943005181347),
 ('fleksibel', 0.031088082901554404),
 ('sulten', 0.031088082901554404)],
16: [('afrika', 0.26800670016750416),
 ('lokal', 0.21273031825795644),
 ('udviklingsbistand', 0.1306532663316583),
 ('bistand', 0.1306532663316583),
```

```
 ('udviklingsminister', 0.07035175879396985),
 ('marie', 0.04187604690117253),
 ('lokalt', 0.04020100502512563),
 ('samarbejdspartner', 0.03685092127303183),
 ('skære', 0.03350083752093802),
 ('næste_åre', 0.018425460636515914)],
17: [('bæredygtig', 0.18315018315018314),
 ('verdensmål', 0.15934065934065933),
 ('partnerskab', 0.15567765567765568),
 ('engagement', 0.08058608058608059),
 ('event', 0.0641025641025641),
 ('innovativ', 0.06043956043956044),
 ('job', 0.06043956043956044),
 ('innovation', 0.05311355311355311),
 ('inspirere', 0.047619047619047616),
 ('udviklingssamarbejde', 0.047619047619047616)],
18: [('tidlig', 0.046511627906976744),
 ('oplever', 0.03875968992248062),
 ('ale', 0.037306201550387594),
 ('fylde', 0.03391472868217054),
 ('altid', 0.03246124031007752),
 ('opleve', 0.03246124031007752),
 ('nedlukning', 0.028585271317829456),
 ('lange', 0.027616279069767442),
 ('viden', 0.027131782945736434),
 ('fond', 0.02131782945736434)],
19: [('være', 0.10583941605839416),
 ('central', 0.08150851581508516),
 ('respons', 0.057177615571776155),
 ('bremse', 0.045012165450121655),
 ('teknologi', 0.032846715328467155),
 ('energi', 0.030413625304136254),
 ('fremhæve', 0.0267639902676399),
 ('verden_leder', 0.0267639902676399),
 ('opbygge', 0.025547445255474453),
 ('sdg', 0.025547445255474453)],
20: [('bruge', 0.19691444600280505),
 ('god', 0.17503506311360448),
 ('flere', 0.14670406732117813),
 ('tid', 0.09200561009817672),
 ('liv', 0.05974754558204769),
 ('stort', 0.057223001402524544),
 ('igen', 0.054137447405329595),
 ('svær', 0.05385694249649369),
 ('nytte', 0.04403927068723703),
 ('finde', 0.04151472650771389)],
21: [('samtale', 0.2682119205298013),
 ('åben', 0.18874172185430463),
 ('resultat', 0.16225165562913907),
 ('bekymring', 0.13245033112582782),
 ('landsindsamling', 0.09271523178807947),
 ('beskrive', 0.06622516556291391),
 ('indsamler', 0.046357615894039736),
 ('forventning', 0.04304635761589404)],
22: [('stå_klare', 0.1186046511627907),
 ('akut_behov', 0.06279069767441861),
 ('enormt', 0.06046511627906977),
 ('psykosocial', 0.05813953488372093),
 ('hele_land', 0.046511627906976744),
 ('pakke', 0.044186046511627906),
 ('førstehjælp', 0.044186046511627906),
```

```
 ('hårdt_bruge', 0.04186046511627907),
 ('bombe', 0.03953488372093023),
 ('udbrud', 0.03953488372093023)],
23: [('direktør', 0.14910025706940874),
 ('kritik', 0.08226221079691516),
 ('uafhængig', 0.055269922879177376),
 ('herunder', 0.05398457583547558),
 ('norge', 0.04627249357326478),
 ('skarp', 0.038560411311053984),
 ('dokumentation', 0.030848329048843187),
 ('lidelse', 0.030848329048843187),
 ('prioritet', 0.028277634961439587),
 ('rasmus', 0.028277634961439587)],
24: [('indsats', 0.25383920505871727),
 ('sårbar', 0.16440831074977416),
 ('grunde', 0.11653116531165311),
 ('næsten', 0.07949412827461608),
 ('påvirke', 0.07588075880758807),
 ('lære', 0.07226738934056007),
 ('nogensinde', 0.05871725383920506),
 ('spiller', 0.03884372177055104),
 ('gøre_forskel', 0.03523035230352303),
 ('hinanden', 0.031616982836495035)],
25: [('fremme', 0.10613397901533494),
 ('lille', 0.09483454398708636),
 ('styrke', 0.08151735270379339),
 ('øge', 0.0811138014527845),
 ('samfund', 0.06658595641646489),
 ('stærk', 0.06335754640839386),
 ('fortsat', 0.05488297013720742),
 ('inden', 0.03631961259079903),
 ('nødvendig', 0.03430185633575464),
 ('vej', 0.03389830508474576)],
26: [('menneskerettighed', 0.2563600782778865),
 ('demokratisk', 0.06555772994129158),
 ('ytringsfrihed', 0.04598825831702544),
 ('menneskerettigheder', 0.03522504892367906),
 ('racisme', 0.03522504892367906),
 ('psykiatri', 0.03131115459882583),
 ('etnisk', 0.03131115459882583),
 ('frihedsrettighed', 0.02837573385518591),
 ('beretning', 0.026418786692759294),
 ('frihed', 0.025440313111545987)],
27: [('klokke', 0.35526315789473684),
 ('deltage', 0.1564327485380117),
 ('dialog', 0.10818713450292397),
 ('lancere', 0.08771929824561403),
 ('forbedre', 0.06871345029239766),
 ('fokusere', 0.05847953216374269),
 ('menneskelig', 0.043859649122807015),
 ('aktør', 0.038011695906432746),
 ('center', 0.03654970760233918),
 ('stikken', 0.03070175438596491)],
28: [('strategi', 0.26055045871559634),
 ('udviklingspolitisk', 0.12293577981651377),
 ('høring', 0.09174311926605505),
 ('on', 0.04954128440366973),
 ('retning', 0.045871559633027525),
 ('tilgang', 0.03853211009174312),
 ('udviklingsarbejde', 0.03669724770642202),
 ('godt_samarbejde', 0.03486238532110092),
```

```
 ('no', 0.03486238532110092),
 ('hen', 0.03302752293577982)],
29: [('positiv', 0.03218727139722019),
 ('gælde', 0.02852962692026335),
 ('slå', 0.02487198244330651),
 ('særlig', 0.021580102414045354),
 ('video', 0.020848573518653987),
 ('ift', 0.018653986832479885),
 ('dårlig', 0.018288222384784197),
 ('europæisk', 0.017922457937088514),
 ('ansætte', 0.017922457937088514),
 ('kronik', 0.017922457937088514)],
30: [('kris', 0.3992443324937028),
 ('løsning', 0.22921914357682618),
 ('stige', 0.21914357682619648),
 ('fælles', 0.15239294710327456)],
31: [('fokus', 0.5709923664122137),
 ('sætte', 0.2687022900763359),
 ('sætte_fokus', 0.08702290076335878),
 ('forandring', 0.0549618320610687),
 ('kø', 0.0183206106870229)],
32: [('sammen', 0.12298387096774194),
 ('komme', 0.10170250896057348),
 ('gøre', 0.09565412186379928),
 ('stå', 0.08086917562724015),
 ('hele', 0.06653225806451613),
 ('kæmpe', 0.06608422939068101),
 ('blive', 0.04704301075268817),
 ('uden', 0.04704301075268817),
 ('første', 0.039874551971326166),
 ('beskytte', 0.03920250896057348)],
33: [('corona', 0.24326672458731538),
 ('politisk', 0.1424847958297133),
 ('coronakrise', 0.1416159860990443),
 ('penge', 0.13292788879235448),
 ('hej', 0.043440486533449174),
 ('privat', 0.035621198957428324),
 ('engagere', 0.030408340573414423),
 ('hænge', 0.02693310165073849),
 ('vinde', 0.026064291920069503),
 ('kunde', 0.026064291920069503)],
34: [('dk', 0.8279158699808795),
 ('forpligtelse', 0.08030592734225621),
 ('alvor', 0.06118546845124283),
 ('dk_burde', 0.030592734225621414)],
35: [('ekstrem', 0.11708860759493671),
 ('skaber', 0.08016877637130802),
 ('investere', 0.06962025316455696),
 ('hårdest', 0.0580168776371308),
 ('dagsorden', 0.04430379746835443),
 ('syde', 0.03270042194092827),
 ('liste', 0.026371308016877638),
 ('verden_befolkning', 0.02531645569620253),
 ('international_solidaritet', 0.024261603375527425),
 ('produktion', 0.020042194092827006)],
36: [('glæde', 0.06801379990142928),
 ('fantastisk', 0.05963528831936915),
 ('dansker', 0.056185312962050274),
 ('glad', 0.045342533267619514),
 ('budskab', 0.04484967964514539),
 ('besøge', 0.04337111877772302),
```

```
 ('snakke', 0.03449975357318876),
 ('videre', 0.03351404632824051),
 ('melde', 0.02661409561360276),
 ('åbne', 0.025628388368654508)],
37: [('civilsamfund', 0.5851063829787234),
 ('mali', 0.14893617021276595),
 ('tilmelding', 0.11347517730496454),
 ('udviklingspolitik', 0.07801418439716312),
 ('super', 0.07446808510638298)],
38: [('udvikling', 0.2612359550561798),
 ('morgen', 0.2148876404494382),
 ('handling', 0.21067415730337077),
 ('hør', 0.1544943820224719),
 ('stemme', 0.09129213483146068),
 ('udnytte', 0.03932584269662921),
 ('jubilæum', 0.028089887640449437)],
39: [('frivillig', 0.4844290657439446),
 ('genbrugsbutik', 0.14186851211072665),
 ('aarhus', 0.10034602076124567),
 ('salg', 0.0657439446366782),
 ('rigtig_god', 0.05536332179930796),
 ('genbrug', 0.05536332179930796),
 ('parat', 0.05190311418685121),
 ('drikke', 0.04498269896193772)],
40: [('coronavirus', 0.23719165085388993),
 ('generalsekretær', 0.2333965844402277),
 ('din', 0.17077798861480076),
 ('donere', 0.0967741935483871),
 ('fordele', 0.056925996204933584),
 ('million_krone', 0.04554079696394687),
 ('fare', 0.04364326375711575),
 ('coronakris', 0.04174573055028463),
 ('avis', 0.030360531309297913),
 ('kollapse', 0.024667931688804556)],
41: [('barn', 1.0)],
42: [('live', 0.21231766612641814),
 ('tilmelde', 0.13776337115072934),
 ('by', 0.09238249594813615),
 ('starter', 0.07293354943273905),
 ('sprede', 0.06320907617504051),
 ('midlertidig', 0.04862236628849271),
 ('stort_tak', 0.04376012965964344),
 ('sektor', 0.04376012965964344),
 ('tackle', 0.03079416531604538),
 ('indflydelse', 0.027552674230145867)],
43: [('hjælpe_barn', 0.12926829268292683),
 ('barn_familie', 0.11219512195121951),
 ('barn_rettighed', 0.1024390243902439),
 ('sårbar_familie', 0.06097560975609756),
 ('din_barn', 0.05121951219512195),
 ('børn', 0.04878048780487805),
 ('barn_liv', 0.046341463414634146),
 ('mio_barn', 0.046341463414634146),
 ('barn_risikere', 0.041463414634146344),
 ('sommerlejr', 0.041463414634146344)],
44: [('land', 0.4731766124171187),
 ('global', 0.22664255575647982),
 ('international', 0.16335141651597349),
 ('lande', 0.07836045810729356),
 ('årti', 0.03134418324291742),
 ('ende', 0.027124773960216998)],
```

```
 45: [('kvinde', 0.5864759427828349),
  ('adgang', 0.3094928478543563),
  ('information', 0.07152145643693109),
  ('vinder', 0.032509752925877766)],
 46: [('dansk', 0.5822550831792976),
  ('opfordre', 0.1266173752310536),
  ('mene', 0.11182994454713494),
  ('skrive', 0.07578558225508318),
  ('faste', 0.06746765249537892),
  ('fri', 0.036044362292051754)],
 47: [('økonomisk', 0.20314389359129384),
  ('system', 0.08585247883917775),
  ('yem', 0.08464328899637243),
  ('økonomi', 0.08222490931076179),
  ('værst', 0.0773881499395405),
  ('ansvarlig', 0.0761789600967352),
  ('milliard', 0.07013301088270858),
  ('udstyre', 0.05562273276904474),
  ('våben', 0.049576783555018135),
  ('redskab', 0.041112454655380895)],
 48: [('ligestilling', 0.7591836734693878),
  ('føde', 0.1346938775510204),
  ('tech', 0.10612244897959183)],
 49: [('stor', 0.17239983812221774),
  ('lige', 0.14447592067988668),
  ('visere', 0.11938486442735735),
  ('kun', 0.09510319708619992),
  ('gange', 0.09307972480777013),
  ('håbe', 0.08781869688385269),
  ('endnu', 0.07891541885876163),
  ('veje', 0.05584783488466208),
  ('tal', 0.043707001214083364),
  ('næste', 0.03885066774585188)],
 50: [('åre', 0.38879645036051025),
  ('siden', 0.11758180809761509),
  ('giver', 0.10815307820299501),
  ('blandt', 0.10537992235163617),
  ('lukke', 0.06544647809206877),
  ('betyde', 0.06211869107043816),
  ('åre_siden', 0.04547975596228508),
  ('begynde', 0.03050471436494731),
  ('starte', 0.021630615640599003),
  ('klog', 0.020521353300055462)],
 51: [('skide', 0.12669683257918551),
  ('angst', 0.12669683257918551),
  ('modig', 0.12217194570135746),
  ('tyskland', 0.11312217194570136),
  ('inkle', 0.10407239819004525),
  ('vold', 0.09049773755656108),
  ('finland', 0.058823529411764705),
  ('absolut', 0.058823529411764705),
  ('specialisere', 0.049773755656108594),
  ('rar', 0.049773755656108594)],
 52: [('samråd', 0.13681592039800994),
  ('retfærdighed', 0.07462686567164178),
  ('afsløre', 0.05970149253731343),
  ('breaking', 0.05223880597014925),
  ('absurd', 0.04975124378109453),
  ('etisk', 0.04477611940298507),
  ('udenrigsminister', 0.04477611940298507),
  ('protest', 0.04228855721393035),
```

```
 ('påpege', 0.04228855721393035),
 ('faktum', 0.03980099502487562)],
53: [('fængsle', 0.24299065420560748),
 ('indsat', 0.11588785046728972),
 ('regime', 0.09158878504672897),
 ('fængsel', 0.08411214953271028),
 ('dokumentere', 0.08037383177570094),
 ('forbrydelse', 0.0616822429906542),
 ('ellebæk', 0.056074766355140186),
 ('undertrykkelse', 0.041121495327102804),
 ('ungarn', 0.037383177570093455),
 ('menneskerettighedskrænkelse', 0.037383177570093455)],
54: [('kina', 0.06745362563237774),
 ('tyrkiet', 0.0657672849915683),
 ('demonstration', 0.048903878583473864),
 ('begå', 0.04215851602023609),
 ('præsident', 0.04215851602023609),
 ('journalist', 0.04047217537942664),
 ('demonstrant', 0.04047217537942664),
 ('kritisere', 0.03541315345699832),
 ('menneskerettighedsforkæmper', 0.03541315345699832),
 ('løslade', 0.03541315345699832)],
55: [('hjælpe', 0.5413461538461538),
 ('behov', 0.3326923076923077),
 ('forskel', 0.09134615384615384),
 ('konference', 0.03461538461538462)],
56: [('menneske', 1.0)],
57: [('vande', 0.25692695214105793),
 ('etiopien', 0.17884130982367757),
 ('indien', 0.08312342569269521),
 ('sæbe', 0.07556675062972293),
 ('hygiejne', 0.0654911838790932),
 ('nøde', 0.06297229219143577),
 ('menneske_verden', 0.04030226700251889),
 ('hjælpe_menneske', 0.04030226700251889),
 ('drikkevand', 0.04030226700251889),
 ('taknemmelig', 0.037783375314861464)],
58: [('lov', 0.2049335863377609),
 ('ulovlig', 0.16698292220113853),
 ('politi', 0.15939278937381404),
 ('straffe', 0.1252371916508539),
 ('dom', 0.0967741935483871),
 ('tilsyn', 0.04933586337760911),
 ('ryste', 0.04174573055028463),
 ('kriminalitet', 0.028462998102466792),
 ('retsstat', 0.024667931688804556),
 ('tarv', 0.024667931688804556)],
59: [('dybt', 0.10235131396957123),
 ('glædelig', 0.07883817427385892),
 ('civil', 0.07745504840940526),
 ('angribe', 0.07053941908713693),
 ('farlig', 0.051175656984785614),
 ('vokse', 0.04426002766251729),
 ('dybt_bekymre', 0.04149377593360996),
 ('hurtigst', 0.034578146611341634),
 ('frygtelig', 0.0318118948824343),
 ('lederskab', 0.029045643153526972)],
60: [('regering', 0.352112676056338),
 ('politiker', 0.09483568075117371),
 ('historisk', 0.061971830985915494),
 ('opråbe', 0.046948356807511735),
```

```
  ('længe', 0.046009389671361506),
  ('aktivist', 0.03943661971830986),
  ('nej', 0.03755868544600939),
  ('kigge', 0.03568075117370892),
  ('brev', 0.03380281690140845),
  ('christiansborg', 0.028169014084507043)],
 61: [('imod', 0.15602836879432624),
  ('statsminister', 0.09397163120567376),
  ('klart', 0.08865248226950355),
  ('mette', 0.08156028368794327),
  ('foran', 0.0797872340425532),
  ('lydere', 0.07801418439716312),
  ('frederiksen', 0.07092198581560284),
  ('mette_frederiksen', 0.05851063829787234),
  ('bede', 0.05319148936170213),
  ('stemmer', 0.04609929078014184)],
 62: [('seksuel', 0.22014051522248243),
  ('krænkelse', 0.1920374707259953),
  ('nette', 0.08196721311475409),
  ('krænke', 0.07728337236533958),
  ('stærkt_bekymre', 0.04449648711943794),
  ('straffelov', 0.04215456674473068),
  ('barn_tarv', 0.02810304449648712),
  ('kæmpe_forskel', 0.02810304449648712),
  ('seksualundervisning', 0.02810304449648712),
  ('ytring', 0.02810304449648712)],
 63: [('rettighed', 0.4655493482309125),
  ('rette', 0.41154562383612664),
  ('online', 0.12290502793296089)],
 64: [('solidaritet', 0.3516483516483517),
  ('ngo', 0.19230769230769232),
  ('støt', 0.16758241758241757),
  ('droppe', 0.12087912087912088),
  ('solidarisk', 0.08241758241758242),
  ('fælles_opråbe', 0.04395604395604396),
  ('erklære', 0.04120879120879121)],
 65: [('godt', 0.34459459459459457),
  ('vigtig', 0.33614864864864863),
  ('afgøre', 0.19932432432432431),
  ('opbakning', 0.060810810810810814),
  ('voks', 0.03462837837837838),
  ('mfle', 0.024493243243243243)],
 66: [('stoppe', 0.39148073022312374),
  ('kræve', 0.31237322515212984),
  ('sidde', 0.1845841784989858),
  ('kær', 0.11156186612576065)],
 67: [('amnesty', 0.1903114186851211),
  ('tyrkisk', 0.11764705882352941),
  ('samtykke', 0.10726643598615918),
  ('amnestys', 0.0657439446366782),
  ('samtykkelov', 0.06228373702422145),
  ('rtc', 0.06228373702422145),
  ('foldschacke', 0.058823529411764705),
  ('advokat', 0.058823529411764705),
  ('knud_foldschacke', 0.058823529411764705),
  ('knud', 0.058823529411764705)],
 68: [('sikre', 0.27994011976047906),
  ('følge', 0.19760479041916168),
  ('give', 0.15494011976047903),
  ('mulighed', 0.1474550898203593),
  ('sat', 0.04940119760479042),
```

```
 ('mand', 0.047904191616766467),
 ('givet', 0.035179640718562874),
 ('køn', 0.035179640718562874),
 ('dermed', 0.026197604790419162),
 ('findes', 0.026197604790419162)],
69: [('fn', 0.3850782190132371),
 ('ekspert', 0.06859205776173286),
 ('true', 0.06738868832731648),
 ('samt', 0.056558363417569195),
 ('torsdag', 0.052948255114320095),
 ('grundlægge', 0.048134777376654635),
 ('in', 0.046931407942238268),
 ('virtuel', 0.046931407942238268),
 ('bane', 0.03850782190132371),
 ('of', 0.030084235860409144)],
70: [('fortæller', 0.30718954248366015),
 ('beskyttelse', 0.22712418300653595),
 ('sikkerhed', 0.1437908496732026),
 ('fejre', 0.09640522875816994),
 ('frygte', 0.08660130718954248),
 ('anledning', 0.07026143790849673),
 ('studie', 0.042483660130718956),
 ('kraftig', 0.026143790849673203)],
71: [('diskutere', 0.08161044613710555),
 ('person', 0.0794341675734494),
 ('eksempel', 0.058759521218716),
 ('maj', 0.05767138193688792),
 ('the', 0.055495103373231776),
 ('overfor', 0.05223068552774755),
 ('vurdere', 0.05223068552774755),
 ('bygge', 0.03808487486398259),
 ('hurtigere', 0.03808487486398259),
 ('virus', 0.03808487486398259)],
72: [('eneste', 0.0976),
 ('mandag', 0.056),
 ('ligesom', 0.048),
 ('fange', 0.048),
 ('coronapandemi', 0.0464),
 ('læge', 0.0464),
 ('nær', 0.0464),
 ('tv', 0.04),
 ('styre', 0.04),
 ('skadelig', 0.0384)],
73: [('arbejder', 0.2668360864040661),
 ('region', 0.07750952986022872),
 ('faso', 0.045743329097839895),
 ('burkina', 0.045743329097839895),
 ('sahele', 0.045743329097839895),
 ('burkina_faso', 0.044472681067344345),
 ('ambassadør', 0.03811944091486658),
 ('world', 0.036848792884371026),
 ('støtte_arbejde', 0.03430749682337993),
 ('kønsbaseret', 0.03430749682337993)],
74: [('asylansøger', 0.198943661971831),
 ('asyl', 0.09154929577464789),
 ('kvoteflygtning', 0.07922535211267606),
 ('flygtning_fordrive', 0.04929577464788732),
 ('grækenland', 0.04929577464788732),
 ('græsk', 0.04401408450704225),
 ('unhcr', 0.04225352112676056),
 ('flygtninge', 0.035211267605633804),
```

```
 ('flygtning_migrant', 0.02992957746478873),
 ('asylbehandling', 0.028169014084507043)],
75: [('danmark', 1.0)],
76: [('tortur', 0.21671018276762402),
 ('autoritær', 0.06527415143603134),
 ('humanrights', 0.05613577023498695),
 ('traumatisere_flygtning', 0.048302872062663184),
 ('dignity', 0.04699738903394256),
 ('libyen', 0.03524804177545692),
 ('torturoffer', 0.03263707571801567),
 ('umenneskelig_behandling', 0.031331592689295036),
 ('traumeramt', 0.028720626631853787),
 ('ydre', 0.024804177545691905)],
77: [('risikere', 0.08670181605155243),
 ('sende', 0.0849443468072642),
 ('to', 0.0849443468072642),
 ('samme', 0.07850029291154072),
 ('netop', 0.07615700058582309),
 ('sen', 0.043936731107205626),
 ('tre', 0.043350908025776215),
 ('svar', 0.03983596953719976),
 ('leve', 0.03690685413005272),
 ('minister', 0.02870533099004101)],
78: [('situation', 0.314785373608903),
 ('tillykke', 0.2273449920508744),
 ('år', 0.22575516693163752),
 ('via', 0.18282988871224165),
 ('drive', 0.0492845786963434)],
79: [('antal', 0.1760355029585799),
 ('risiko', 0.17455621301775148),
 ('forebygge', 0.11686390532544379),
 ('tvinge', 0.09911242603550297),
 ('lide', 0.08579881656804733),
 ('forværre', 0.0650887573964497),
 ('far', 0.05917159763313609),
 ('fire', 0.051775147928994084),
 ('daglig', 0.04437869822485207),
 ('langvarig', 0.03254437869822485)],
80: [('flugte', 0.20365535248041775),
 ('fordrive', 0.10966057441253264),
 ('flygtningelejr', 0.10966057441253264),
 ('flygte', 0.10574412532637076),
 ('menneske_flugte', 0.06657963446475196),
 ('migration', 0.044386422976501305),
 ('afghanistan', 0.03655352480417755),
 ('hoved', 0.033942558746736295),
 ('østafrika', 0.033942558746736295),
 ('brand', 0.030026109660574413)],
81: [('verden', 1.0)],
82: [('tag', 0.1266944734098019),
 ('lytte', 0.09697601668404589),
 ('høre', 0.07247132429614181),
 ('lad', 0.06047966631908238),
 ('ekstra', 0.05735140771637122),
 ('råde', 0.056308654848800835),
 ('lave', 0.053180396246089674),
 ('først', 0.043274244004171014),
 ('fortælle', 0.04171011470281543),
 ('tusind', 0.038060479666319084)],
83: [('hjem', 0.3918757467144564),
 ('syrien', 0.23416965352449223),
```

```
 ('syrisk', 0.2031063321385902),
 ('lejr', 0.11230585424133811),
 ('voldtægt', 0.03942652329749104),
 ('standse', 0.019115890083632018)],
84: [('folketing', 0.15281899109792285),
 ('borger', 0.12759643916913946),
 ('aldersdiskrimination', 0.10089020771513353),
 ('bud', 0.07566765578635015),
 ('handlingsplan', 0.05341246290801187),
 ('forsker', 0.040059347181008904),
 ('facebook', 0.032640949554896145),
 ('simpelthen', 0.028189910979228485),
 ('forbi', 0.028189910979228485),
 ('stod', 0.028189910979228485)],
85: [('grænse', 0.17407407407407408),
 ('behandle', 0.10925925925925926),
 ('respekt', 0.07037037037037037),
 ('flygtningebarn', 0.05925925925925926),
 ('sikkert', 0.053703703703703705),
 ('opholdstilladelse', 0.05),
 ('læsse_indlæg', 0.044444444444444446),
 ('moralsk', 0.03148148148148148),
 ('forestille', 0.03148148148148148),
 ('finte', 0.03148148148148148)],
86: [('demokrati', 0.6157205240174672),
 ('spændende', 0.24017467248908297),
 ('princip', 0.14410480349344978)],
87: [('levere', 0.1391614629794826),
 ('sidste', 0.13380909901873328),
 ('mål', 0.10258697591436218),
 ('sidste_åre', 0.06868867082961641),
 ('undgå', 0.057091882247992866),
 ('tro', 0.057091882247992866),
 ('skabe', 0.057091882247992866),
 ('voldsom', 0.05530776092774309),
 ('ifølge', 0.043710972346119537),
 ('betaler', 0.039250669045495096)],
88: [('sundpol', 0.482926829268292 67),
 ('isolation', 0.33170731707317075),
 ('misbruge', 0.11219512195121951),
 ('værdige', 0.07317073170731707)],
89: [('fredelig', 0.34710743801652894),
 ('militær', 0.3305785123966942),
 ('is', 0.15702479338842976),
 ('abort', 0.09090909090909091),
 ('udtalelse', 0.0743801652892562)],
90: [('værre', 0.07207207207207207),
 ('læg', 0.06126126126126126),
 ('selvfølgelig', 0.05945945945945946),
 ('opmærksomhed', 0.05405405405405406),
 ('foregå', 0.04504504504504504),
 ('offentliggøre', 0.04144144144144144),
 ('gribe', 0.03963963963963964),
 ('amerikansk', 0.03963963963963964),
 ('prøve', 0.036036036036036036),
 ('læsning', 0.036036036036036036)],
91: [('stille', 0.10315985130111524),
 ('artikel', 0.06784386617100371),
 ('analyse', 0.055762081784386616),
 ('konkret', 0.05483271375464684),
 ('leder', 0.05111524163568773),
```

```
 ('ambition', 0.03810408921933085),
 ('forklare', 0.0362453531598513),
 ('lægge', 0.03531598513011153),
 ('den', 0.031598513011152414),
 ('omfatte', 0.03066914498141264)],
92: [('eu', 0.7905604719764012),
 ('eupol', 0.13274336283185842),
 ('par', 0.07669616519174041)],
93: [('anden', 0.2408906882591093),
 ('have', 0.13157894736842105),
 ('blandt_anden', 0.11538461538461539),
 ('kritisk', 0.09919028340080972),
 ('alligevel', 0.08704453441295547),
 ('københavn', 0.08704453441295547),
 ('senere', 0.07489878542510121),
 ('afstand', 0.06477732793522267),
 ('gruppe', 0.058704453441295545),
 ('tilgængelig', 0.04048582995951417)],
94: [('lovforslag', 0.16013071895424835),
 ('eudk', 0.1437908496732026),
 ('human', 0.10457516339869281),
 ('høringssvar', 0.10457516339869281),
 ('tredjeland', 0.08823529411764706),
 ('bo', 0.07516339869281045),
 ('arbejdsmarked', 0.0718954248366013),
 ('danmark_burde', 0.06535947712418301),
 ('modtagecenter', 0.058823529411764705),
 ('tværtimod', 0.0457516339869281)],
95: [('moria', 0.2774566473988439),
 ('tag_ansvar', 0.15606936416184972),
 ('tag_imod', 0.12138728323699421),
 ('kvoteflygtninge', 0.10982658959537572),
 ('demo', 0.10404624277456648),
 ('flygtningepolitik', 0.09248554913294797),
 ('usolidarisk', 0.06936416184971098),
 ('frigive', 0.06936416184971098)],
96: [('borgerforslag', 0.26540284360189575),
 ('ms', 0.13270142180094788),
 ('oliejagt', 0.10900473933649289),
 ('nordsøen', 0.08056872037914692),
 ('støt_borgerforslag', 0.061611374407582936),
 ('stoppe_oliejagt', 0.05687203791469194),
 ('coskat', 0.05687203791469194),
 ('oliejagt_nordsøen', 0.052132701421800945),
 ('klimaråde', 0.04739336492890995),
 ('afgifte', 0.04739336492890995)],
97: [('johanne', 0.14385964912280702),
 ('schmidtnielsen', 0.10877192982456141),
 ('johanne_schmidtnielsen', 0.10877192982456141),
 ('rede', 0.07894736842105263),
 ('rede_barn', 0.07368421052631578),
 ('grooming', 0.0456140350877193),
 ('skriver_johanne', 0.03508771929824561),
 ('sørensen', 0.03333333333333333),
 ('seksuel_overgreb', 0.03333333333333333),
 ('overgreb_barn', 0.02456140350877193)],
98: [('hente', 0.15217391304347827),
 ('barn_hjem', 0.1358695652173913),
 ('hente_barn', 0.04891304347826087),
 ('barn_mor', 0.04891304347826087),
 ('lejr_syrien', 0.03804347826086957),
```

```
 ('grov', 0.028985507246376812),
 ('syrisk_lejre', 0.028985507246376812),
 ('barn_levere', 0.028985507246376812),
 ('hjem_syrisk', 0.028985507246376812),
 ('fangelejr_syrien', 0.028985507246376812)],
99: [('dansk_barn', 0.3402985074626866),
 ('fangelejr', 0.2656716417910448),
 ('syrisk_fangelejr', 0.14626865671641792),
 ('barn_syrisk', 0.06865671641791045),
 ('hente_hjem', 0.056716417910447764),
 ('alroj', 0.04776119402985075),
 ('alhol', 0.041791044776119404),
 ('bringbørnenehjem', 0.03283582089552239)],
100: [('sag', 0.2646638054363376),
 ('bekymre', 0.2145922746781116),
 ('rådgivning', 0.07296137339055794),
 ('lovgivning', 0.07296137339055794),
 ('skade', 0.0715307582260372),
 ('spørge', 0.045779685264663805),
 ('dømme', 0.044349070100143065),
 ('praksis', 0.044349070100143065),
 ('svigte', 0.044349070100143065),
 ('opmærksom', 0.04005722460658083)],
101: [('pige', 0.38962472406181015),
 ('pige_kvinde', 0.05187637969094923),
 ('barn_flugte', 0.03090507726269316),
 ('børneægteskab', 0.029801324503311258),
 ('mv', 0.02869757174392936),
 ('hkh', 0.02759381898454746),
 ('adgang_ren', 0.025386313465783666),
 ('pige_rettighed', 0.025386313465783666),
 ('gifte', 0.025386313465783666),
 ('ung_pige', 0.019867549668874173)],
102: [('alvorlig', 0.3231441048034934),
 ('udsætte', 0.2554585152838428),
 ('psykisk', 0.24017467248908297),
 ('ofre', 0.13755458515283842),
 ('særligt_udsat', 0.043668122270742356)],
103: [('konsekvens', 0.23017902813299232),
 ('bidrage', 0.20545609548167093),
 ('fremtid', 0.18073316283034954),
 ('hurtig', 0.09207161125319693),
 ('gå', 0.06223358908780904),
 ('langsigtet', 0.05200341005967604),
 ('konflikte', 0.04859335038363171),
 ('skrøbelig', 0.0392156862745098),
 ('tværs', 0.03495311167945439),
 ('stigning', 0.03154305200341006)],
104: [('million', 0.7748344370860927),
 ('million_barn', 0.17660044150110377),
 ('fordoble', 0.04856512141280353)],
105: [('dkmedier', 0.9213973799126638), ('dræbe', 0.07860262008733625)],
106: [('miste', 0.13316261203585147),
 ('samtidig', 0.10499359795134443),
 ('hånd', 0.09475032010243278),
 ('ødelægge', 0.088348271446863),
 ('fortsætte', 0.07554417413572344),
 ('måske', 0.07554417413572344),
 ('enorm', 0.058898847631242),
 ('korte', 0.052496798975672214),
 ('vælge', 0.052496798975672214),
```

```
 ('udsat_menneske', 0.03713188220230474)],
107: [('kollega', 0.18682634730538922),
 ('hårdt', 0.17604790419161676),
 ('smitte', 0.09101796407185629),
 ('usa', 0.08023952095808383),
 ('begrænse', 0.07784431137724551),
 ('cirka', 0.05149700598802395),
 ('trussel', 0.05149700598802395),
 ('oplysning', 0.046706586826347304),
 ('heldigvis', 0.04311377245508982),
 ('basal', 0.03592814371257485)],
108: [('vilkår', 0.15158924205378974),
 ('emne', 0.1100244498777506),
 ('tanke', 0.10024449877750612),
 ('understøtte', 0.05867970660146699),
 ('relevant', 0.05378973105134474),
 ('inddragelse', 0.044009779951100246),
 ('faglig', 0.039119804400977995),
 ('usikker', 0.03667481662591687),
 ('sagsbehandling', 0.03667481662591687),
 ('sikre_god', 0.03178484107579462)],
109: [('stede', 0.3298059964726631),
 ('læse', 0.14109347442680775),
 ('ønske', 0.13051146384479717),
 ('lede', 0.1128747795414462),
 ('interviewe', 0.09171075837742504),
 ('værdi', 0.08818342151675485),
 ('tilbyde', 0.06525573192239859),
 ('systematisk', 0.04056437389770723)],
110: [('måtte', 0.2656800563777308),
 ('få', 0.13953488372093023),
 ('ingen', 0.12825933756166313),
 ('hvorfor', 0.10570824524312897),
 ('problem', 0.1014799154334038),
 ('tage', 0.09372797744890768),
 ('se', 0.05637773079633545),
 ('grade', 0.052854122621564484),
 ('hvem', 0.02959830866807611),
 ('vigtigste', 0.026779422128259338)],
111: [('tilbage', 0.2807017543859649),
 ('overgreb', 0.13225371120107962),
 ('årig', 0.1039136302294197),
 ('erfaring', 0.08906882591093117),
 ('lejre', 0.08097165991902834),
 ('forvejen', 0.06612685560053981),
 ('vende', 0.0620782726045884),
 ('rådgive', 0.044534412955465584),
 ('vender', 0.03643724696356275),
 ('vende_tilbage', 0.02834008097165992)],
112: [('migrant', 0.4166666666666667),
 ('afvise', 0.26515151515151514),
 ('udbrede', 0.17424242424242425),
 ('minimum', 0.14393939393939395)],
113: [('behandling', 0.16404199475065617),
 ('traumatisere', 0.10761154855643044),
 ('juridisk', 0.08136482939632546),
 ('traume', 0.08005249343832022),
 ('dkkrim', 0.06430446194225722),
 ('ptsd', 0.06167979002624672),
 ('umenneskelig', 0.06036745406824147),
 ('ekspertise', 0.06036745406824147),
```

```
 ('patient', 0.05905511811023622),
 ('europaråd', 0.03937007874015748)],
114: [('alder', 0.0676056338028169),
 ('panele', 0.0676056338028169),
 ('februar', 0.0676056338028169),
 ('kvindelig', 0.0676056338028169),
 ('rykke', 0.0676056338028169),
 ('gravid', 0.0676056338028169),
 ('ung_kvinde', 0.061971830985915494),
 ('8marts', 0.059154929577464786),
 ('tegne', 0.056338028169014086),
 ('idé', 0.05352112676056338)],
115: [('egen', 0.18164435946462715),
 ('lav', 0.14149139579349904),
 ('stolt', 0.13766730401529637),
 ('rest', 0.10516252390057361),
 ('19', 0.08221797323135756),
 ('forfærdelig', 0.08221797323135756),
 ('glemme', 0.07265774378585087),
 ('sige', 0.055449330783938815),
 ('lørdag', 0.055449330783938815),
 ('ordentlig', 0.05353728489483748)],
116: [('vaccine', 0.5681818181818182),
 ('patent', 0.12987012987012986),
 ('sygeplejerske', 0.06168831168831169),
 ('fordeling', 0.06168831168831169),
 ('producer', 0.05194805194805195),
 ('patent_vaccine', 0.03571428571428571),
 ('vaccine_verden', 0.032467532467532464),
 ('adgang_vaccine', 0.032467532467532464),
 ('covax', 0.025974025974025976)],
117: [('rapport', 0.5410094637223974),
 ('før', 0.305993690851735),
 ('spørgsmål', 0.1529968454258675)],
118: [('linke', 0.30246913580246915),
 ('trist', 0.2345679012345679),
 ('stilling', 0.18518518518518517),
 ('vaccinere', 0.1728395061728395),
 ('oplyse', 0.10493827160493827)],
119: [('indlæg', 0.42081447963800905),
 ('forhold', 0.20361990950226244),
 ('medlem', 0.19909502262443438),
 ('politik', 0.17647058823529413)],
120: [('ramme', 0.2853566958698373),
 ('løfte', 0.13141426783479349),
 ('understrege', 0.06382978723404255),
 ('ambitiøs', 0.05381727158948686),
 ('tilfælde', 0.05006257822277847),
 ('info', 0.03754693366708386),
 ('hårdt_ram', 0.03629536921151439),
 ('hus', 0.03629536921151439),
 ('pointe', 0.03629536921151439),
 ('regne', 0.03379224030037547)],
121: [('klimabistand', 0.15837937384898712),
 ('klimatilpasning', 0.09023941068139964),
 ('dkimpact', 0.055248618784530384),
 ('klimaudsat', 0.055248618784530384),
 ('vejre', 0.053406998158379376),
 ('klimastøtte', 0.049723756906077346),
 ('dag_klimatal', 0.04788213627992634),
 ('klimatal', 0.04788213627992634),
```

```
 ('ekstrem_vejre', 0.04419889502762431),
 ('care', 0.04419889502762431)],
122: [('skriver', 1.0)],
123: [('flygtning', 0.7842778793418648),
 ('syrisk_flygtning', 0.07861060329067641),
 ('nærområde', 0.06946983546617916),
 ('syrer', 0.06764168190127971)],
124: [('mor', 0.4807692307692308),
 ('more', 0.25),
 ('adskille', 0.07051282051282051),
 ('sex', 0.07051282051282051),
 ('kontekst', 0.0641025641025641),
 ('tak_gode', 0.0641025641025641)],
125: [('barn_ung', 0.1527777777777778),
 ('forælder', 0.08935185185185185),
 ('udsat_barn', 0.062037037037037036),
 ('trivsel', 0.05925925925925926),
 ('skolechat', 0.04861111111111111),
 ('voksen', 0.04537037037037037),
 ('anbringe', 0.030555555555555555),
 ('tilbage_skole', 0.022222222222222223),
 ('anbringe_barn', 0.018518518518518517),
 ('føler', 0.017592592592592594)],
126: [('akut', 0.13082039911308205),
 ('dø', 0.1164079822616408),
 ('katastrofe', 0.1164079822616408),
 ('middel', 0.10421286031042129),
 ('redde', 0.07649667405764966),
 ('prise', 0.06097560975609756),
 ('verden_stor', 0.06097560975609756),
 ('overleve', 0.041019955654102),
 ('humanitær_katastrofe', 0.036585365853658534),
 ('katastrofal', 0.02993348115299335)],
127: [('indsamling', 0.26903553299492383),
 ('donation', 0.15228426395939088),
 ('såre', 0.1319796954314721),
 ('hjemmeside', 0.1065989847715736),
 ('indsamle', 0.09644670050761421),
 ('landechef', 0.08629441624365482),
 ('mobilepay', 0.08629441624365482),
 ('lægehjælp', 0.07106598984771574)],
128: [('offentlig', 0.2318840579710145),
 ('institution', 0.09782608695652174),
 ('lønne', 0.09057971014492754),
 ('opgøre', 0.07971014492753623),
 ('problematisk', 0.07246376811594203),
 ('åbenhed', 0.06884057971014493),
 ('analysere', 0.057971014492753624),
 ('skandale', 0.05434782608695652),
 ('techgigant', 0.05434782608695652),
 ('oprindelig', 0.05434782608695652)],
129: [('ung', 1.0)],
130: [('menneskeret', 0.213089802130898),
 ('dkhandicap', 0.0943683409436834),
 ('retssikkerhed', 0.0806697108066971),
 ('institut', 0.0700152207001522),
 ('chikane', 0.060882800608828),
 ('statsborgerskab', 0.0471841704718417),
 ('institut_menneskerettighed', 0.0471841704718417),
 ('seksuel_chikane', 0.0350076103500761),
 ('hadforbrydelse', 0.0334855403348554),
```

```
 ('menneskeretlig', 0.0319634703196347)],
131: [('barn_vilkår', 0.09879518072289156),
 ('stopsvigt', 0.09397590361445783),
 ('anbringelse', 0.06506024096385542),
 ('skolefravær', 0.05783132530120482),
 ('børnefaglig', 0.05301204819277108),
 ('børnetelefon', 0.05060240963855422),
 ('digidk', 0.04819277108433735),
 ('faglighed', 0.04578313253012048),
 ('underretning', 0.04578313253012048),
 ('konsulent', 0.043373493975903614)],
132: [('myndighed', 0.2261072261072261),
 ('rejse', 0.15384615384615385),
 ('offer', 0.15151515151515152),
 ('ret', 0.13752913752913754),
 ('fysisk', 0.10023310023310024),
 ('overholde', 0.07692307692307693),
 ('pol', 0.055944055944055944),
 ('opholde', 0.03496503496503497),
 ('brutal', 0.03496503496503497),
 ('sikkerhedsstyrke', 0.027972027972027972)],
133: [('social', 0.4086206896551724),
 ('debat', 0.3),
 ('anbefaling', 0.16034482758620688),
 ('regel', 0.07931034482758621),
 ('input', 0.05172413793103448)],
134: [('hospital', 0.18385650224215247),
 ('klinik', 0.09417040358744394),
 ('lgbt', 0.08520179372197309),
 ('forfølgelse', 0.08520179372197309),
 ('flygtningenævn', 0.06726457399103139),
 ('fattiger', 0.053811659192825115),
 ('bomber', 0.053811659192825115),
 ('løbende', 0.053811659192825115),
 ('asylcenter', 0.053811659192825115),
 ('asylansøger_flygtning', 0.04932735426008968)],
135: [('familie', 0.84375), ('tak_støtte', 0.15625)],
136: [('dksocial', 1.0)],
137: [('undersøgelse', 0.12783751493428913),
 ('kommune', 0.08721624850657109),
 ('mediere', 0.08124253285543608),
 ('handicap', 0.06212664277180406),
 ('anbefale', 0.06212664277180406),
 ('undersøge', 0.05734767025089606),
 ('social_mediere', 0.054958183990442055),
 ('dktech', 0.037037037037037035),
 ('indhold', 0.037037037037037035),
 ('folkeskole', 0.02986857825567503)],
138: [('dr', 0.39361702127659576),
 ('grønland', 0.3829787234042553),
 ('botilbud', 0.11702127659574468),
 ('plejehjem', 0.10638297872340426)],
139: [('ensomhed', 0.5131086142322098),
 ('mistrivsel', 0.1947565543071161),
 ('bupartnerskab', 0.09737827715355805),
 ('overgang', 0.052434456928838954),
 ('telefon', 0.052434456928838954),
 ('hjælp_udsat', 0.04494382022471910),
 ('tilbage_fællesskab', 0.04494382022471910)],
140: [('fattigdom', 0.215210355987055),
 ('virksomhed', 0.18446601941747573),
```

```
 ('mia', 0.16990291262135923),
 ('mødes', 0.0970873786407767),
 ('tråde', 0.0744336569579288),
 ('tjene', 0.061488673139158574),
 ('halvdel', 0.059870550161812294),
 ('årlig', 0.03398058252427184),
 ('new', 0.030744336569579287),
 ('redder', 0.024271844660194174)],
141: [('vente', 0.09714889123548047),
 ('for_eksempel', 0.07497360084477296),
 ('uddpol', 0.05807814149947202),
 ('forstå', 0.048574445617740235),
 ('per', 0.046462513199577615),
 ('kontrol', 0.04012671594508976),
 ('slette', 0.03801478352692714),
 ('lærer', 0.03590285110876452),
 ('skyld', 0.0337909186906019),
 ('visere_rapport', 0.03167898627243928)],
142: [('digital', 0.45454545454545453),
 ('oplevelse', 0.12039312039312039),
 ('dele', 0.11793611793611794),
 ('ressource', 0.10319410319410319),
 ('udspille', 0.10073710073710074),
 ('ane', 0.05405405405405406),
 ('hensyn', 0.04914004914004914)],
143: [('uddannelse', 0.3295238095238095),
 ('krig', 0.3219047619047619),
 ('sundhed', 0.2704761904761905),
 ('hel_afgøre', 0.07809523809523809)],
144: [('formand', 0.12596401028277635),
 ('gammel', 0.11568123393316196),
 ('aktivitet', 0.10796915167095116),
 ('sted', 0.10282776349614396),
 ('præsentere', 0.09511568123393316),
 ('medicin', 0.08740359897172237),
 ('fulde_gange', 0.07969151670951156),
 ('isolere', 0.06940874035989718),
 ('modvirke', 0.05141388174807198),
 ('løb', 0.04884318766066838)],
145: [('dkcivil', 0.22522522522522523),
 ('korse', 0.15090090090090091),
 ('rød', 0.1373873873873874),
 ('rød_korse', 0.11711711711711711),
 ('ensom', 0.07657657657657657),
 ('folkemøde', 0.04054054054054054),
 ('tilstede', 0.038288288288288286),
 ('behov_hjælp', 0.036036036036036036),
 ('idlibe', 0.02927927927927928),
 ('hvid', 0.02702702702702703)],
146: [('skole', 0.6144366197183099),
 ('elevere', 0.07394366197183098),
 ('beskytte_barn', 0.058098591549295774),
 ('sikre_barn', 0.04929577464788732),
 ('barn_skole', 0.04225352112676056),
 ('give_barn', 0.035211267605633804),
 ('involvere', 0.028169014084507043),
 ('fungere', 0.02640845070422535),
 ('konsekvens_barn', 0.02640845070422535),
 ('barn_åre', 0.022887323943661973)],
147: [('ren', 0.162227602905569),
 ('bor', 0.12106537530266344),
```

```
 ('verden_udsat', 0.08958837772397095),
 ('ren_vande', 0.08958837772397095),
 ('udviklingspolitisk_strategi', 0.08958837772397095),
 ('sundhedssystem', 0.07021791767554479),
 ('hæve', 0.04600484261501211),
 ('gt', 0.0387409200968523),
 ('sanitet', 0.03631961259079903),
 ('ren_drikkevand', 0.031476997578692496)],
148: [('alkohol', 0.2198581560283688),
 ('hjemløs', 0.2198581560283688),
 ('blå', 0.1773049645390071),
 ('aldersgrænse', 0.1276595744680851),
 ('blå_korse', 0.09219858156028368),
 ('alkoholkultur', 0.09219858156028368),
 ('ungdomsuddannelse', 0.07092198581560284)],
149: [('mio', 0.6820175438596491), ('ram', 0.31798245614035087)],
150: [('krone', 0.45087719298245615),
 ('mio_krone', 0.156140350877193),
 ('nødhjælp', 0.15087719298245614),
 ('købe', 0.07192982456140351),
 ('overskud', 0.06666666666666667),
 ('indtægt', 0.03684210526315789),
 ('nordlig', 0.03684210526315789),
 ('jorden', 0.02982456140350877)],
151: [('klimakrise', 0.18527315914489312),
 ('love', 0.14014251781472684),
 ('skriv', 0.1163895486935867),
 ('olie', 0.09263657957244656),
 ('co', 0.0855106888361045),
 ('gas', 0.040380047505938245),
 ('kur', 0.03800475059382423),
 ('fossil', 0.03800475059382423),
 ('kule', 0.035629453681710214),
 ('grønt', 0.0332541567695962)],
152: [('fællesskab', 0.825), ('omsorg', 0.175)],
153: [('dkgreen', 0.8193548387096774),
 ('klimalov', 0.05),
 ('reduktion', 0.04516129032258064),
 ('grøn_retfærdig', 0.03387096774193549),
 ('klimapolitik', 0.02903225806451613),
 ('stopoliejagten', 0.02258064516129032)],
154: [('klima', 0.2376068376068376),
 ('grøn', 0.2376068376068376),
 ('klimahandling', 0.08888888888888889),
 ('omstilling', 0.08376068376068375),
 ('parisaftale', 0.05982905982905983),
 ('grøn_omstilling', 0.05641025641025641),
 ('klimakris', 0.035897435897435895),
 ('uland', 0.034188034188034191),
 ('klimamål', 0.02905982905982906),
 ('foregangsland', 0.023931623931623933)],
155: [('myanmar', 0.8481012658227848), ('militærkuppe', 0.1518987341772152)],
156: [('webinar', 0.24464831804281345),
 ('dkbiz', 0.22935779816513763),
 ('valg', 0.19571865443425077),
 ('investering', 0.18654434250764526),
 ('genopbygning', 0.06116207951070336),
 ('bni', 0.04281345565749235),
 ('klimaindsats', 0.039755351681957186)],
157: [('bekæmpe', 0.2521891418563923),
 ('befolkning', 0.2486865148861646),
```

```
 ('folk', 0.2084063047285464),
 ('hjælpepakke', 0.07355516637478109),
 ('libanon', 0.07180385288966724),
 ('løs', 0.0647985989492119),
 ('midt', 0.06304728546409807),
 ('holland', 0.017513134851138354)],
158: [('rig', 0.4474327628361858),
 ('retfærdig', 0.1784841075794621),
 ('rig_land', 0.14180929095354522),
 ('selskab', 0.11735941320293398),
 ('fair', 0.11491442542787286)],
159: [('caritum', 0.24647887323943662),
 ('udviklingspolitisk_humanitær', 0.14084507042253522),
 ('humanitær_strategi', 0.11971830985915492),
 ('global_fokus', 0.11267605633802817),
 ('danmission', 0.1056338028169014),
 ('no_on', 0.09859154929577464),
 ('gf', 0.09154929577464789),
 ('folkekirken', 0.08450704225352113)],
160: [('uganda', 0.3282051282051282),
 ('kenya', 0.1641025641025641),
 ('gaza', 0.14871794871794872),
 ('ude_verden', 0.12307692307692308),
 ('skrue', 0.1076923076923077),
 ('højtryk', 0.07692307692307693),
 ('arbejder_højtryk', 0.05128205128205128)],
161: [('eksplosion', 0.13877551020408163),
 ('beirutte', 0.12244897959183673),
 ('udsende', 0.12244897959183673),
 ('lesbo', 0.09387755102040816),
 ('morialejr', 0.07755102040816327),
 ('telte', 0.06938775510204082),
 ('kirke', 0.053061224489795916),
 ('hjælpeorganisation', 0.04081632653061224),
 ('ilte', 0.04081632653061224),
 ('mio_flygtning', 0.04081632653061224)],
162: [('fattig', 0.5384615384615384),
 ('verden_fattig', 0.2865761689291101),
 ('fattig_land', 0.17496229260935142)],
163: [('19dk', 0.40522875816993464),
 ('tilbud', 0.19607843137254902),
 ('barn_voksen', 0.13071895424836602),
 ('hjælpe_udsat', 0.10457516339869281),
 ('bus', 0.08496732026143791),
 ('børnefamilie', 0.0784313725490196)],
164: [('chef', 0.5416666666666666),
 ('bestyrelse', 0.2),
 ('who', 0.16666666666666666),
 ('dkjob', 0.09166666666666666)],
165: [('klimaforandring', 0.23366013071895425),
 ('mio_menneske', 0.08823529411764706),
 ('oversvømmelse', 0.08823529411764706),
 ('bazar', 0.06372549019607843),
 ('tørke', 0.06372549019607843),
 ('asie', 0.06209150326797386),
 ('bangladesh', 0.06209150326797386),
 ('coxs', 0.058823529411764705),
 ('coxs_bazar', 0.05718954248366013),
 ('lockdown', 0.03758169934640523)],
166: [('ulighed', 0.5683760683760684),
 ('banke', 0.1346153846153846),
```

```
 ('dansk_banke', 0.0641025641025641),
 ('gæld', 0.05982905982905983),
 ('ulighedsrapport', 0.049145299145299144),
 ('nordea', 0.038461538461538464),
 ('dkfinans', 0.029914529914529916),
 ('milliardær', 0.029914529914529916),
 ('banke_nordea', 0.02564102564102564)],
167: [('nordisk', 0.10841121495327102),
 ('reducere', 0.08598130841121496),
 ('lokalsamfund', 0.08224299065420561),
 ('opnå', 0.07476635514018691),
 ('april', 0.052336448598130844),
 ('kontor', 0.04299065420560748),
 ('praktikant', 0.03925233644859813),
 ('projektere', 0.03925233644859813),
 ('formål', 0.03364485981308411),
 ('rundt_verden', 0.03364485981308411)],
168: [('skattely', 0.1669902912621359),
 ('skatte', 0.08640776699029126),
 ('oxfam', 0.08058252427184466),
 ('ibis', 0.04854368932038835),
 ('oxfam_ibis', 0.04660194174757282),
 ('verden_rig', 0.03300970873786408),
 ('pensionskasse', 0.02233009708737864),
 ('terma', 0.02233009708737864),
 ('skattelyliste', 0.020388349514563107),
 ('krig_yem', 0.019417475728155338)],
169: [('betale', 0.4034090909090909),
 ('mia_krone', 0.26704545454545453),
 ('bank', 0.20454545454545456),
 ('løbe', 0.125)],
170: [('sulte', 0.22981366459627328),
 ('million_menneske', 0.14285714285714285),
 ('hungersnød', 0.03975155279503106),
 ('madspild', 0.03602484472049689),
 ('sydsudan', 0.02981366459627329),
 ('spise', 0.02857142857142857),
 ('ernæring', 0.02732919254658385),
 ('fødevarehjælp', 0.02360248447204969),
 ('marked', 0.02236024844720497),
 ('dkk', 0.02236024844720497)],
171: [('udviklingsland', 0.3053435114503817),
 ('reaktion', 0.09923664122137404),
 ('irak', 0.07633587786259542),
 ('skarpt', 0.07633587786259542),
 ('dansk_ngo', 0.05343511450381679),
 ('årligt', 0.05343511450381679),
 ('fremlægge', 0.05343511450381679),
 ('guide', 0.05343511450381679),
 ('indenfor', 0.05343511450381679),
 ('somalia', 0.05343511450381679)],
172: [('verdensmålene', 0.11072664359861592),
 ('glip', 0.07958477508650519),
 ('læ', 0.07612456747404844),
 ('multilateral', 0.0726643598615917),
 ('socioøkonomisk', 0.06228373702422145),
 ('multilateralisme', 0.058823529411764705),
 ('genopretning', 0.058823529411764705),
 ('undp', 0.04844290657439446),
 ('dansk_tid', 0.04844290657439446),
 ('indvirkning', 0.04498269896193772)],
```

```
 173: [('butik', 0.1951219512195122),
  ('wefood', 0.15447154471544716),
  ('klimahjælp', 0.15447154471544716),
  ('etc', 0.11382113821138211),
  ('folkeret', 0.10569105691056911),
  ('klimahjælp_verden', 0.0975609756097561),
  ('folkekirke', 0.08943089430894309),
  ('william', 0.08943089430894309)],
 174: [('psykolog', 0.43548387096774194),
  ('mental', 0.3010752688172043),
  ('mental_sundhed', 0.1774193548387097),
  ('sårbar_barn', 0.08602150537634409)],
 175: [('unicef', 0.3353115727002967),
  ('verden_barn', 0.10682492581602374),
  ('talomverdensbørn', 0.0771513353115727),
  ('norden', 0.05934718100890208),
  ('kime', 0.05637982195845697),
  ('nepal', 0.05341246290801187),
  ('unicef_danmark', 0.050445103857566766),
  ('kime_hartznvære', 0.04154302670623145),
  ('hartznvære', 0.04154302670623145),
  ('jobdk', 0.03560830860534125)]}
```

In [ ]:

Appendix 19. Part of speech

Installing all packets

In [1]:

```
# Setup spacy
#! pip install -U pip setuptools wheel
#! pip install -U spacy
#! python -m spacy download da_core_news_sm
#!pip install altair

import spacy
import pandas as pd
import re
import nltk
from collections import Counter
import networkx as nx
import matplotlib.pyplot as plt
import altair as alt
import numpy as np

sp_danish = spacy.load("da_core_news_sm") #We use the danish packets, as our words are Dani
```

In [5]:

```
import pandas as pd
df = pd.read_csv('data_danish.csv')
df
```

Out[5]:

	actor	tweet	date	retweet	date_convert	@mentions
0	PlanBornefonden	13-årige Larissa bor i Sahel-regionen, og var ...	2021-05-14 09:03:00	NaN	2021-05-14 00:00:00	NaN
1	PlanBornefonden	Vi ønsker alle muslimer en god Eid i aften! Ei...	2021-05-12 14:00:02	NaN	2021-05-12 00:00:00	NaN
2	PlanBornefonden	Kom til samtalekøkken med @BosseStine og @Clau...	2021-05-12 11:58:03	@dorthe10	2021-05-12 00:00:00	['BosseStine ClausMeyerDK'
3	PlanBornefonden	Mali, Burkina Faso og Niger - også kendt som d...	2021-05-12 10:00:02	NaN	2021-05-12 00:00:00	NaN
4	PlanBornefonden	Vores seje kollega Iben Østergaard Markussen f...	2021-05-12 09:23:14	@dorthe10	2021-05-12 00:00:00	['radioloud_dk MaternityF'
...	...	...	...	...	...	..
9444	CaritasDanmark	Vores Internationale chef Betina Gollander-Jen...	2020-03-03 13:46:14	NaN	2020-03-03 00:00:00	NaN
9445	CaritasDanmark	#askeonsdag \nFasteindsamling https://t.co/...	2020-02-26 12:26:26	NaN	2020-02-26 00:00:00	['Pesitho1'
9446	CaritasDanmark	Dolores Halpin-Bachmann har arbejdet for #Cari...	2020-02-07 10:49:47	NaN	2020-02-07 00:00:00	NaN
9447	CaritasDanmark	Børn skal være børn - https://t.co/h5xNWh7a7U ...	2020-01-31 13:48:55	NaN	2020-01-31 00:00:00	NaN
9448	CaritasDanmark	”Ansvaret hviler i høj grad på jeres generatio...	2020-01-21 12:00:47	NaN	2020-01-21 00:00:00	NaN

9449 rows × 16 columns

Using spacey to categorize words

In [6]:

```
%time df['nlp_spacy'] = df.clean_text.apply(sp_danish)
```

```
Wall time: 2min 13s
```

In [7]:

```
typ2common = {}
for tweet in df.nlp_spacy:
    for w in tweet:
        typ = w.pos_
        if not typ in typ2common:
            typ2common[typ] = Counter()
        typ2common[typ][w.lemma_]+=1
```

In [8]:

```
#here we see all the different word types, and the top 10 used words in these types.

most_used = []
for typ in sorted(typ2common,key=lambda x: sum(typ2common[x].values()),reverse=True):
    most = typ,typ2common[typ].most_common(20) #we take the most common 10 words
    most_used.append(most)
    print(most)
```

```
('NOUN', [('barn', 2515), ('dag', 1262), ('verden', 1151), ('år', 884), ('la
nd', 875), ('tak', 804), ('menneske', 712), ('danmark', 562), ('brug', 562),
('læ', 472), ('kvinde', 450), ('hjælp', 426), ('flygtning', 412), ('støtte',
407), ('arbejde', 398), ('regering', 365), ('fokus', 362), ('pige', 348),
('million', 348), ('rapport', 346)])
('ADP', [('i', 11113), ('til', 6120), ('for', 5905), ('af', 4155), ('på', 40
60), ('mede', 3839), ('om', 2787), ('fra', 1999), ('som', 1834), ('mod', 71
9), ('over', 512), ('unde', 502), ('vide', 406), ('ende', 394), ('efter', 34
8), ('blandt', 190), ('uden', 177), ('hos', 161), ('mellem', 129), ('genne
m', 113)])
('VERB', [('være', 1402), ('have', 1106), ('få', 949), ('se', 760), ('sige',
651), ('komme', 601), ('hjælpe', 544), ('gå', 455), ('give', 444), ('gøre',
426), ('støtte', 423), ('skrive', 422), ('stå', 379), ('vise', 363), ('sætt
e', 354), ('sikre', 322), ('ramme', 312), ('arbejde', 301), ('gø', 270), ('f
ortælle', 265)])
('ADV', [('der', 2744), ('ikke', 1957), ('nu', 1158), ('her', 1094), ('så',
904), ('også', 771), ('hvor', 661), ('mere', 619), ('hvordan', 551), ('samme
n', 546), ('mede', 511), ('ud', 480), ('oppe', 429), ('lige', 354), ('derfo
r', 330), ('til', 307), ('helt', 289), ('mest', 287), ('godte', 277), ('hje
m', 273)])
('ADJ', [('ny', 1009), ('stor', 842), ('vigtig', 795), ('alle', 712), ('mang
en', 650), ('dansk', 616), ('unge', 613), ('god', 530), ('flere', 523), ('me
re', 383), ('udsætte', 362), ('fattig', 360), ('global', 321), ('hele', 29
2), ('gode', 281), ('humanitær', 265), ('klare', 254), ('afgøre', 223), ('st
ærk', 217), ('social', 194)])
('AUX', [('være', 6287), ('have', 2221), ('kunne', 2029), ('skulle', 1627),
('blive', 1090), ('ville', 786), ('måtte', 366), ('bør', 366), ('hav', 58),
('burde', 47), ('nå', 9), ('tø', 9), ('mv', 8), ('rtc', 6), ('dk', 5), ('aku
t', 5), ('turde', 5), ('tilbyde', 4), ('autoritær', 4), ('eus', 3)])
('PRON', [('vi', 4766), ('det', 3246), ('de', 1382), ('du', 798), ('jeg', 58
6), ('sig', 544), ('man', 381), ('hvad', 311), ('nogen', 291), ('den', 225),
('selv', 214), ('en', 210), ('hver', 205), ('anden', 200), ('hun', 190), ('h
an', 159), ('hvilken', 108), ('hvis', 66), ('denne', 62), ('ingen', 58)])
('DET', [('en', 4740), ('de', 1956), ('vores', 1664), ('den', 1456), ('det',
915), ('deres', 755), ('denne', 447), ('anden', 338), ('sin', 326), ('min',
165), ('nogen', 146), ('din', 142), ('ingen', 124), ('hendes', 60), ('eus',
56), ('dit', 55), ('jeres', 44), ('hans', 39), ('and', 34), ('én', 33)])
('CCONJ', [('og', 10278), ('men', 978), ('eller', 374), ('samt', 47), ('fo
r', 6), ('dr', 4), ('køn', 3), ('ml', 2), ('sm', 2), ('ps', 2), ('p+', 1),
('værsgo', 1), ('un', 1), ('jeps', 1), ('it', 1), ('minoritet', 1)])
('PART', [('at', 5893), ('dk', 8), ('mens', 3), ('fordi', 1), ('maraton',
1), ('taals', 1), ('nikolaj', 1), ('mundbind', 1), ('lam', 1)])
('SCONJ', [('at', 1603), ('nå', 497), ('hvis', 426), ('fordi', 129), ('men
s', 120), ('da', 68), ('så', 64), ('føre', 62), ('om', 40), ('selvom', 36),
('ligesom', 27), ('indtil', 9), ('covid', 8), ('til', 8), ('inden', 7), ('me
dmindre', 5), ('hvornår', 4), ('imens', 4), ('side', 3), ('hvorvidt', 3)])
('X', [('kl', 169), ('pga', 101), ('dk', 86), ('m', 79), ('læ', 69), ('wfp',
64), ('the', 49), ('nå', 36), ('s', 29), ('ibis', 25), ('of', 24), ('mia', 2
3), ('co', 22), ('and', 21), ('world', 19), ('corona', 19), ('\u2060', 18),
('dr', 17), ('syrien', 13), ('ift', 13)])
```

```
('SPACE', [(' ', 1447), ('  ', 1)])
('NUM', [('to', 145), ('tre', 74), ('fem', 48), ('fire', 35), ('tie', 29),
('dk', 27), ('seks', 19), ('akut', 17), ('mv', 16), ('ni', 15), ('syv', 15),
('otte', 9), ('eus', 8), ('no', 6), ('kl', 6), ('rohingyaer', 5), ('morias',
4), ('vm', 4), ('kontantoverførsler', 4), ('gt', 3)])
('PROPN', [('danmarks', 84), ('danmark', 53), ('europa', 38), ('usa', 33),
('grækenland', 26), ('|', 23), ('p', 19), ('s', 15), ('afghanistan', 13),
('ekspertise', 11), ('grønlands', 7), ('syrien', 7), ('sn', 7), ('betina',
7), ('michael', 6), ('denmark', 6), ('henrik', 4), ('poul', 4), ('dr', 4),
('online', 4)])
('INTJ', [('dk', 72), ('ja', 44), ('tillykke', 23), ('nej', 4), ('corona',
3), ('åh', 2), ('etc', 1), ('feje', 1), ('ok', 1), ('afrika', 1), ('tråd',
1), ('mee', 1)])
('SYM', [('=', 34), ('ml', 3), ('t', 3), ('lgbt', 3), ('+', 1), ('test', 1),
('elected', 1), ('drummond', 1), ('``', 1)])
('PUNCT', [('kl', 7), ('dr', 1), ('pct', 1), ('indlæg', 1)])
```

Top 10 most used Nouns, verbs and adj

In [9]:

```
#We here make a function that can give us dataframe of the word types we would like to look

def hapser(word_list):
    word_list = word_list[1]
    words = []
    values = []

    for a, b in (word_list):
        words.append(a)
        values.append(b)

    d = {'Words' : words, 'Values' : values}
    df = pd.DataFrame(data=d)

    return df
```

In [10]:

```
#Here we see all the word types and their indexes. We are interested in Nouns, Verbs, Adver
#We can therefore see their indexes here, and use those with our function.
index = 0
for value in most_used:
    print(index, value[0])
    index += 1
```

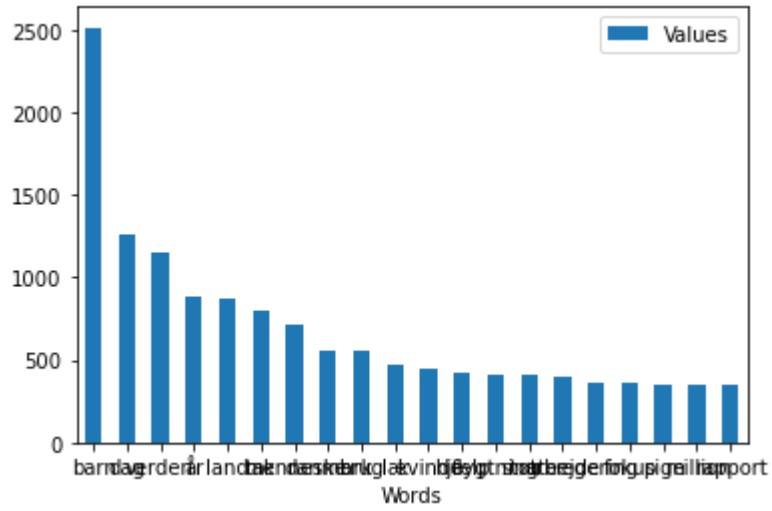
```
0 NOUN
1 ADP
2 VERB
3 ADV
4 ADJ
5 AUX
6 PRON
7 DET
8 CCONJ
9 PART
10 SCONJ
11 X
12 SPACE
13 NUM
14 PROPN
15 INTJ
16 SYM
17 PUNCT
```

In [11]:

```
#Here we define the most used nouns and use our function to show them as dataframe
nouns = most_used[0]
verbs = most_used[2]
adjectives = most_used[4]
proper_nouns = most_used[14]

#here we make them all into dataframes
df_nouns = hapser(nouns)
df_verbs = hapser(verbs)
df_adj = hapser(adjectives)
df_propn = hapser(proper_nouns)
```

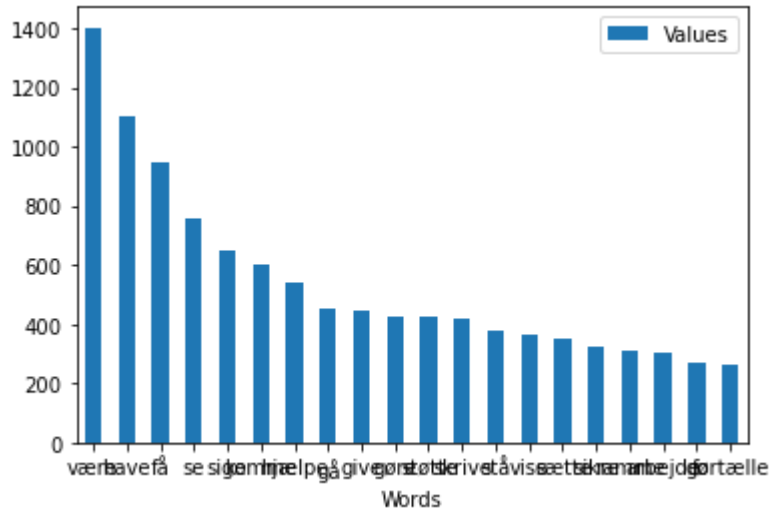
In [12]:

```
ax_nouns = df_nouns.plot.bar(x='Words', y='Values', rot=0)
```



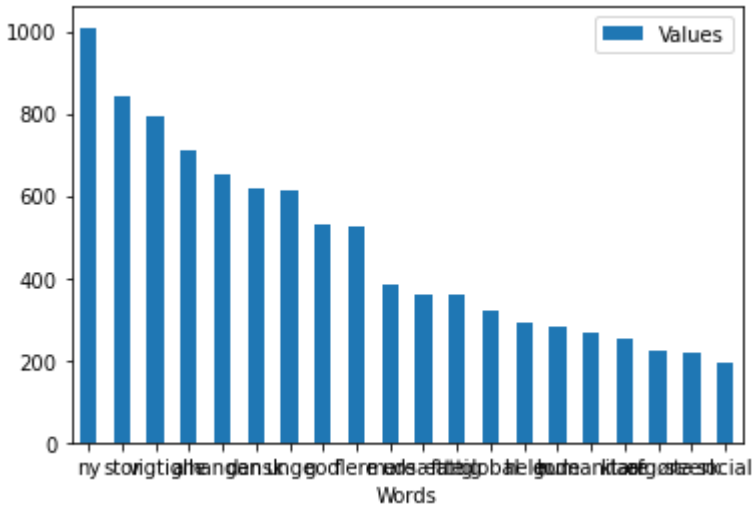
In [13]:

```
ax_verbs = df_verbs.plot.bar(x='Words', y='Values', rot=0)
```



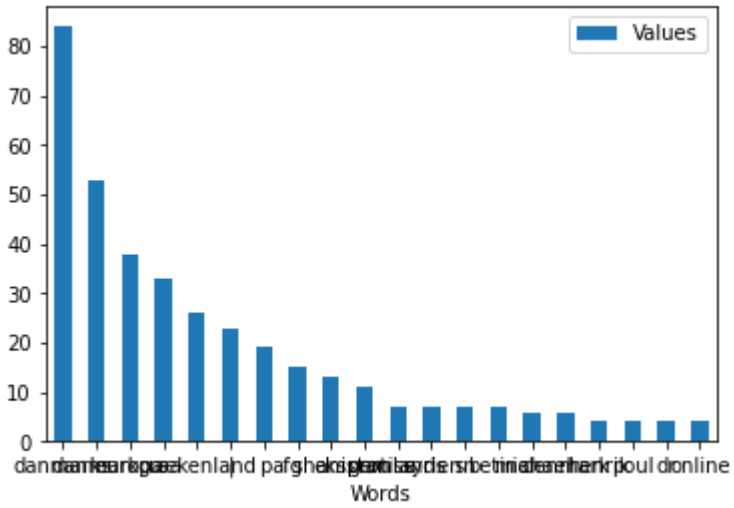
In [14]:

```
ax_adj = df_adj.plot.bar(x='Words', y='Values', rot=0)
```



In [15]:

```
ax_propn = df_propn.plot.bar(x='Words', y='Values', rot=0)
```



Top used word types per NGO

In [17]:

```
tweets_agg = df.groupby(['actor'], as_index = False).agg({'tweet': ' '.join,'clean_text': '
tweets_agg
```

Out[17]:

	actor	tweet	clean_text
0	ADRA_Danmark	Humanitær støtte til #Syrien sker i tæt samarb...	humanitær støtte til sker i tæt samarbejde med...
1	AVestegnen	Tænk at dine børn hopper på trampolin, mens de...	tænk at dine børn hopper på trampolin mens der...
2	ActionAidDK	Det er endelig blevet tid til klimamarch igen!...	det er endelig blevet tid til klimamarch igen ...
3	BornsVilkar	Gode tiltag i udspil fra @regeringDK, men savn...	gode tiltag i udspil fra men savner fokus på f...
4	CARE_Danmark	@CARE fejrer 75års fødselsdag\n\nCARE blev ska...	fejrer års fødselsdag care blev skabt i for a...
5	CaritasDanmark	"Brug magt med visdom"\nHør vores generalsekre...	brug magt med visdom hør vores generalsekretær...
6	DKIndsamling	♡👏 Se med klokken 19.00! 👏♡ \n\n#lillelands...	se med klokken hej igen det er forkert fns f...
7	DRC_dk	Fra jan-april forhindrede europæiske myndighed...	fra janapril forhindrede europæiske myndighede...
8	DignityDK	Danske skatteydere finansierer den såkaldte li...	danske skatteydere finansierer den såkaldte li...
9	MissionEast	Nepal er det nye Indien. FN kalder på akut ind...	nepal er det nye indien fn kalder på akut inds...
10	PlanBornefonden	13-årige Larissa bor i Sahel-regionen, og var ...	årige larissa bor i sahelregionen og var i tvu...
11	RTC_DK	Jeg ved godt at verden er kompleks, men jeg fa...	jeg ved godt at verden er kompleks men jeg fat...
12	RefWelcome	Forvirret over syrernes 3 forsk. asylstatus, d...	forvirret over syrernes forsk asylstatus det u...
13	UNDP_Danmark	Det er vigtigere end nogensinde, at vi står sa...	det er vigtigere end nogensinde at vi står sam...
14	UNICEFDK	Kan du koordinere pressearbejde, sociale medie...	kan du koordinere pressearbejde sociale medier...
15	WFP_DK	”Det er en bimlende absurditet, at 690 million...	det er en bimlende absurditet at millioner men...
16	amnestydk	Israels angreb mod beboelseskvarterer i #Gaza ...	israels angreb mod beboelseskvarterer i skal u...
17	blaakorsdanmark	Flere kæmper med misbrug, psykisk mistrivsel o...	flere kæmper med misbrug psykisk mistrivsel og...
18	danmissiondk	.@danmissiondk, @CaritasDanmark, BaptistKirken...	baptistkirken og chin christian association d...
19	danskrodekors	Det frivillige sociale engagement i DK er stor...	det frivillige sociale engagement i dk er stor...
20	globaltfokus	HvisDKønsker at medskabe bæredygtig global for...	hvisønsker at medskabe bæredygtig global foran...

	actor	tweet	clean_text
21	lgbt_asylum	Vi ses til demo om lidt! Vi slår ring om syrer...	vi ses til demo om lidt vi slår ring om syrern...
22	menneskeret	Hvordan bekæmper vi corona uden at gå på kompr...	hvordan bekæmper vi corona uden at gå på kompr...
23	msf_dk	Hvis der ikke er ilt, dør kritiske syge #COVID...	hvis der ikke er ilt dør kritiske syge patient...
24	noedhjaelp	Tak for den store omsorg og opbakning ifm den ...	tak for den store omsorg og opbakning ifm den ...
25	oxfamibis	Vi er dybt bekymrede over situationen i Jerusa...	vi er dybt bekymrede over situationen i jerusa...
26	redbarnetdk	Børnene er altid de mest udsatte – også i konf	børnene er altid de mest udsatte også i konfli

In [21]:

```
tweets_agg['nlp_spacy'] = tweets_agg.clean_text.apply(sp_danish)
tweets_agg
```

Out[21]:

	actor	tweet	clean_text	nlp_spacy
0	ADRA_Danmark	Humanitær støtte til #Syrien sker i tæt samarb...	humanitær støtte til sker i tæt samarbejde med...	(humanitær, støtte, til, sker, i, tæt, samarbe...
1	AVestegnen	Tænk at dine børn hopper på trampolin, mens de...	tænk at dine børn hopper på trampolin mens der...	(tænk, at, dine, børn, hopper, på, trampolin, ...
2	ActionAidDK	Det er endelig blevet tid til klimamarch igen!...	det er endelig blevet tid til klimamarch igen ...	(det, er, endelig, blevet, tid, til, klimamarc...
3	BornsVilkar	Gode tiltag i udspil fra @regeringDK, men savn...	gode tiltag i udspil fra men savner fokus på f...	(gode, tiltag, i, udspil, fra, men, savner, fo...
4	CARE_Danmark	@CARE fejrer 75års fødselsdag\n\nCARE blev ska...	fejrer års fødselsdag care blev skabt i for a...	(, fejrer, års, fødselsdag, care, blev, skabt...
5	CaritasDanmark	"Brug magt med visdom"\nHør vores generalsekre...	brug magt med visdom hør vores generalsekretær...	(brug, magt, med, visdom, hør, vores, generals...
6	DKIndsamling	♡👏 Se med klokken 19.00! 👏♡ \n\n#lillelands...	se med klokken hej igen det er forkert fns f...	(, se, med, klokken, , hej, igen, det, er, f...
7	DRC_dk	Fra jan-april forhindrede europæiske myndighed...	fra janapril forhindrede europæiske myndighede...	(fra, janapril, forhindrede, europæiske, myndi...
8	DignityDK	Danske skatteydere finansierer den såkaldte li...	danske skatteydere finansierer den såkaldte li...	(danske, skatteydere, finansierer, den, såkald...
9	MissionEast	Nepal er det nye Indien. FN kalder på akut ind...	nepal er det nye indien fn kalder på akut inds...	(nepal, er, det, nye, indien, fn, kalder, på, ...
10	PlanBornefonden	13-årige Larissa bor i Sahel-regionen, og var ...	årige larissa bor i sahelregionen og var i tvu...	(årige, larissa, bor, i, sahelregionen, og, va...
11	RTC_DK	Jeg ved godt at verden er kompleks, men jeg fa...	jeg ved godt at verden er kompleks men jeg fat...	(jeg, ved, godt, at, verden, er, kompleks, men...
12	RefWelcome	Forvirret over syrernes 3 forsk. asylstatus, d...	forvirret over syrernes forsk asylstatus det u...	(forvirret, over, syrernes, forsk, asylstatus,...
13	UNDP_Danmark	Det er vigtigere end nogensinde, at vi står sa...	det er vigtigere end nogensinde at vi står sam...	(det, er, vigtigere, end, nogensinde, at, vi, ...
14	UNICEFDK	Kan du koordinere pressearbejde, sociale medie...	kan du koordinere pressearbejde sociale medier...	(kan, du, koordinere, pressearbejde, sociale, ...
15	WFP_DK	”Det er en bimlende absurditet, at 690 million...	det er en bimlende absurditet at millioner men...	(det, er, en, bimlende, absurditet, at, millio...
16	amnestydk	Israels angreb mod beboelseskvarterer i #Gaza ...	israels angreb mod beboelseskvarterer i skal u...	(israels, angreb, mod, beboelseskvarterer, i, ...

	actor	tweet	clean_text	nlp_spacy
17	blaakorsdanmark	Flere kæmper med misbrug, psykisk mistrivsel o...	flere kæmper med misbrug psykisk mistrivsel og...	(flere, kæmper, med, misbrug, psykisk, mistriv...
18	danmissiondk	.@danmissiondk, @CaritasDanmark, BaptistKirken...	baptistkirken og chin christian association d...	(, baptistkirken, og, chin, christian, associ...
19	danskrodekors	Det frivillige sociale engagement i DK er stor...	det frivillige sociale engagement i dk er stor...	(det, frivillige, sociale, engagement, i, dk, ...
20	globaltfokus	Hvisᴅᴋønsker at medskabe bæredygtig global for...	hvisønsker at medskabe bæredygtig global foran...	(hvisønsker, at, medskabe, bæredygtig, global,...
21	lgbt_asylum	Vi ses til demo om lidt! Vi slår ring om syrer...	vi ses til demo om lidt vi slår ring om syrern...	(vi, ses, til, demo, om, lidt, vi, slår, ring,...
22	menneskeret	Hvordan bekæmper vi corona uden at gå på kompr...	hvordan bekæmper vi corona uden at gå på kompr...	(hvordan, bekæmper, vi, corona, uden, at, gå, ...
23	msf_dk	Hvis der ikke er ilt, dør kritiske syge #COVID...	hvis der ikke er ilt dør kritiske syge patient...	(hvis, der, ikke, er, ilt, dør, kritiske, syge...
24	noedhjaelp	Tak for den store omsorg og opbakning ifm den ...	tak for den store omsorg og opbakning ifm den ...	(tak, for, den, store, omsorg, og, opbakning, ...
25	oxfamibis	Vi er dybt bekymrede over situationen i Jerusa...	vi er dybt bekymrede over situationen i jerusa...	(vi, er, dybt, bekymrede, over, situationen, i...
26	redbarnetdk	Børnene er altid de mest udsatte – også i konf...	børnene er altid de mest udsatte også i konfli...	(børnene, er, altid, de, mest, udsatte, også, ...

In [22]:

```
typ3common = {"Actor":[], "NOUN":[], "VERB":[], "ADJ":[], "TOTAL":[]}
for i in range (len(tweets_agg)):
    typ3common["Actor"].append(tweets_agg.actor[i])
    nouns = verbs = adjs =  0
    for tweet in tweets_agg.nlp_spacy[i]:
        typ = tweet.pos_
        if typ == "NOUN":
            nouns+=1
        if typ == "VERB":
            verbs+=1
        if typ == "ADJ":
            adjs+=1

    #Populate with data
    total = nouns+verbs+adjs
    typ3common["TOTAL"].append(total)

    typ3common["NOUN"].append(round(float(nouns/total * 100)))
    typ3common["VERB"].append(round(float(verbs/total * 100)))
    typ3common["ADJ"].append(round(float(adjs/total * 100)))


df_words = pd.DataFrame.from_dict(typ3common)
df_words = df_words.sort_values(by=['NOUN'])
df_words
```

Out[22]:

	Actor	NOUN	VERB	ADJ	TOTAL
3	BornsVilkar	51	27	22	10147
19	danskrodekors	52	25	23	6077
2	ActionAidDK	53	29	19	5887
17	blaakorsdanmark	53	23	23	1964
12	RefWelcome	53	28	19	802
26	redbarnetdk	54	26	20	11058
23	msf_dk	54	28	18	2056
4	CARE_Danmark	54	26	19	6909
7	DRC_dk	54	27	18	4570
16	amnestydk	54	28	18	5706
25	oxfamibis	54	26	21	14874
11	RTC_DK	54	27	19	1962
21	lgbt_asylum	55	30	15	1258
13	UNDP_Danmark	55	25	20	6085
8	DignityDK	55	24	20	12789
15	WFP_DK	56	27	17	9902
24	noedhjaelp	56	24	20	6082
10	PlanBornefonden	56	24	20	6073

	Actor	NOUN	VERB	ADJ	TOTAL
14	UNICEFDK	57	26	18	5800
18	danmissiondk	57	27	16	882
20	globaltfokus	57	23	20	2461
22	menneskeret	57	25	19	8855
1	AVestegnen	57	25	18	2067
5	CaritasDanmark	58	21	21	951
9	MissionEast	59	25	17	1454
0	ADRA_Danmark	61	14	24	49

In [23]:

```
typ3common = {"Actor":[], "NOUN":[], "VERB":[], "ADJ":[], "ELSE_WORD":[], "TOTAL":[]}
for i in range (len(tweets_agg)):
    typ3common["Actor"].append(tweets_agg.actor[i])
    nouns = verbs = adjs = else_word = 0
    for tweet in tweets_agg.nlp_spacy[i]:
        typ = tweet.pos_
        if typ == "NOUN":
            nouns+=1
        if typ == "VERB":
            verbs+=1
        if typ == "ADJ":
            adjs+=1
        else:
            else_word+=1

    #Populate with data
    total = nouns+verbs+adjs+else_word
    typ3common["TOTAL"].append(total)

    typ3common["NOUN"].append(round(float(nouns/total * 100)))
    typ3common["VERB"].append(round(float(verbs/total * 100)))
    typ3common["ADJ"].append(round(float(adjs/total * 100)))
    typ3common["ELSE_WORD"].append(round(float(else_word/total * 100)))

df_words = pd.DataFrame.from_dict(typ3common)
df_words
```

Out[23]:

	Actor	NOUN	VERB	ADJ	ELSE_WORD	TOTAL
0	ADRA_Danmark	22	5	9	64	135
1	AVestegnen	21	9	7	64	5686
2	ActionAidDK	18	10	6	66	17570
3	BornsVilkar	18	9	7	66	29636
4	CARE_Danmark	19	9	7	64	19279
5	CaritasDanmark	21	8	8	64	2619
6	DKIndsamling	23	8	4	65	240
7	DRC_dk	19	9	6	66	13358
8	DignityDK	20	9	7	64	35774
9	MissionEast	21	9	6	64	4025
10	PlanBornefonden	20	8	7	65	17367
11	RTC_DK	19	9	7	66	5692
12	RefWelcome	18	9	6	67	2423
13	UNDP_Danmark	20	9	7	64	16950
14	UNICEFDK	20	9	6	65	16535
15	WFP_DK	20	9	6	65	27900
16	amnestydk	19	10	6	65	16219
17	blaakorsdanmark	19	8	8	65	5621

	Actor	NOUN	VERB	ADJ	ELSE_WORD	TOTAL
18	danmissiondk	21	10	6	64	2473
19	danskrodekors	18	9	8	65	17429
20	globaltfokus	20	8	7	64	6851
21	lgbt_asylum	18	10	5	67	3849
22	menneskeret	20	9	7	64	24858
23	msf_dk	18	10	6	66	5995
24	noedhjaelp	20	9	7	65	17190
25	oxfamibis	19	9	7	65	42270
26	redbarnetdk	19	9	7	65	31366

In [24]:

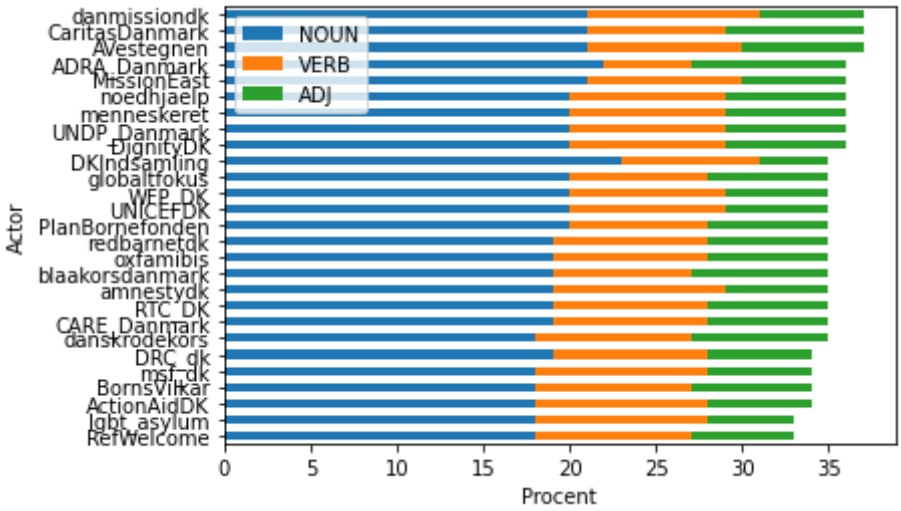
```
df_words['Total_three'] = df_words['NOUN'] + df_words['VERB'] + df_words['ADJ']
df_words = df_words.sort_values(by=['Total_three', 'NOUN'])
df_words
```

Out[24]:

	Actor	NOUN	VERB	ADJ	ELSE_WORD	TOTAL	Total_three
12	RefWelcome	18	9	6	67	2423	33
21	lgbt_asylum	18	10	5	67	3849	33
2	ActionAidDK	18	10	6	66	17570	34
3	BornsVilkar	18	9	7	66	29636	34
23	msf_dk	18	10	6	66	5995	34
7	DRC_dk	19	9	6	66	13358	34
19	danskrodekors	18	9	8	65	17429	35
4	CARE_Danmark	19	9	7	64	19279	35
11	RTC_DK	19	9	7	66	5692	35
16	amnestydk	19	10	6	65	16219	35
17	blaakorsdanmark	19	8	8	65	5621	35
25	oxfamibis	19	9	7	65	42270	35
26	redbarnetdk	19	9	7	65	31366	35
10	PlanBornefonden	20	8	7	65	17367	35
14	UNICEFDK	20	9	6	65	16535	35
15	WFP_DK	20	9	6	65	27900	35
20	globaltfokus	20	8	7	64	6851	35
6	DKIndsamling	23	8	4	65	240	35
8	DignityDK	20	9	7	64	35774	36
13	UNDP_Danmark	20	9	7	64	16950	36
22	menneskeret	20	9	7	64	24858	36
24	noedhjaelp	20	9	7	65	17190	36
9	MissionEast	21	9	6	64	4025	36
0	ADRA_Danmark	22	5	9	64	135	36
1	AVestegnen	21	9	7	64	5686	37
5	CaritasDanmark	21	8	8	64	2619	37
18	danmissiondk	21	10	6	64	2473	37

In [25]:

```
df_words_vis = df_words.drop(['ELSE_WORD', 'TOTAL', 'Total_three'], axis=1)
df_words_vis.set_index("Actor",drop=True,inplace=True)

df_words_vis.plot(kind='barh', stacked=True)
plt.xlabel("Procent")
plt.ylabel("Actor")
plt.savefig('Pct_words')
```



In [ ]:

Appendix 20. Classifier

In [2]:

```
import pandas as pd
import re
import matplotlib.pyplot as plt

#Machine learning packages
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression

#Packages to create DFM
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer

#Packages for cross-validation and parameter tuning
from sklearn.model_selection import StratifiedKFold
from sklearn.model_selection import GridSearchCV
from sklearn.pipeline import Pipeline

#Packages for getting model performance metrics
from sklearn.metrics import plot_confusion_matrix
from sklearn.metrics import classification_report
from sklearn.metrics import roc_curve, auc
from sklearn.metrics import roc_auc_score
```

In [3]:

```
data = pd.read_csv('data_danish.csv', index_col=False)
data.head(3)
```

Out[3]:

	actor	tweet	date	retweet	date_convert	@mentions	#hashtags
0	PlanBornefonden	13-årige Larissa bor i Sahel-regionen, og var ...	2021-05-14 09:03:00	NaN	2021-05-14 00:00:00	NaN	NaN
1	PlanBornefonden	Vi ønsker alle muslimer en god Eid i aften! Ei...	2021-05-12 14:00:02	NaN	2021-05-12 00:00:00	NaN	NaN
2	PlanBornefonden	Kom til samtalekøkken med @BosseStine og @Clau...	2021-05-12 11:58:03	@dorthe10	2021-05-12 00:00:00	['BosseStine, ClausMeyerDK']	dkfooc

In [4]:

```
data.proc_text_all[2]
```

Out[4]:

```
'komme samtalekøkken maj spise lækker ret menu græskinspirere men klog kvind
elighed kærlighed penge magte komme_samtalekøkken samtalekøkken_maj maj_spis
e spise_lækker lækker_ret ret_menu menu_græskinspirere græskinspirere_men me
n_klog klog_kvindelighed kvindelighed_kærlighed kærlighed_penge penge_magte
dkfood '
```

In [3]:

```
#use proc_text_all
critism_string = 'kalder had_forbryder forskel regeringen samme mattiastesfaye mattia
human_string = 'yde assistance engagement forandringer indsatser udvikling arbejde c
```

In [4]:

```
from nltk.tokenize import TweetTokenizer
tknzr = TweetTokenizer()

critism = tknzr.tokenize(critism_string)
human = tknzr.tokenize(human_string)
```

In [5]:

```
len(data)*0.2
```

Out[5]:

```
1889.8000000000002
```

In [6]:

```
list_of_tokens = []
for string in data['proc_text_all']:
    item = tknzr.tokenize(string)
    list_of_tokens.append(item)

data['tokens'] = list_of_tokens
```

In [7]:

```
#count how many words find in the respective lists

critism_n = []
for words in data['tokens']: #this shall be tokens
    i = 0
    for token in words:
        if token in critism:
            i += 1
    critism_n.append(i)

data['critism'] = critism_n

#If equal number of words, drop
human_n = []
for words in data['tokens']: #this shall be tokens
    i = 0
    for token in words:
        if token in human:
            i += 1
    human_n.append(i)

data['human'] = human_n
```

In [8]:

```
data.head(2)
```

Out[8]:

	actor	tweet	date	retweet	date_convert	@mentions	#hashtags	emojis
0	PlanBornefonden	13-årige Larissa bor i Sahel-regionen, og var ...	2021-05-14 09:03:00	NaN	2021-05-14 00:00:00	NaN	NaN	
1	PlanBornefonden	Vi ønsker alle muslimer en god Eid i aften! Ei...	2021-05-12 14:00:02	NaN	2021-05-12 00:00:00	NaN	NaN	

In [9]:

```
#Has the tweets most words from human or critism? Or equal number of words

class_type = []

for i, row in data.iterrows():
    if row.critism > row.human:
        class_type.append('critism')
    elif row.human > row.critism:
        class_type.append('human')
    else:
        class_type.append('equal')

data['class'] = class_type
```

In [10]:

```
data['class_type'] = data['class'].apply(lambda x: 1 if x == 'critism' else 0)


#Creating a test dataset
unlabeled = data.sample(n = round((len(data)*0.1)), axis = 0)
unlabeled = unlabeled.reset_index(drop=True)

#creating test dataset
labeled = data
labeled = data.drop(unlabeled.index, axis=0)

print('Shape of labeled dataset:',labeled.shape,
      '\nShape of unlabeled dataset:',unlabeled.shape)
```

```
Shape of labeled dataset: (8504, 21)
Shape of unlabeled dataset: (945, 21)
```

In [11]:

```
#If equal number of words, drop
print('Before dropping equal classes:', len(labeled))
labeled = labeled[labeled['class'] != 'equal']

print('After dropping equal classes:', len(labeled))
```

```
Before dropping equal classes: 8504
After dropping equal classes: 6476
```

In [12]:

```
#Splitting features into X and y
X = labeled['proc_text_all']
y = labeled['class_type'].values
```

Supervised Learning: Lasso

In [13]:

```
#Initializing the pipeline
pipeline = Pipeline([
    ('vect', CountVectorizer(ngram_range = (1,2), max_df = 0.999, min_df = 0.01, tokenizer
    ('tfidf', TfidfTransformer()),
    ('lasso', LogisticRegression(penalty = 'l1', solver = 'saga', max_iter = 1000)),
])
```

In [14]:

```
#Setting the parameter grid
parameter_grid = {
    'tfidf__use_idf': [True, False],
    'lasso__C': [0.05, 0.1, 0.5, 1, 5]}

#Initializing a kfold with 5 folds
cv = StratifiedKFold(n_splits=5)

lasso_search = GridSearchCV(pipeline, parameter_grid, cv=cv, verbose=10)
```

In [15]:

```
lasso_result = lasso_search.fit(X, y)

#lasso_result.best_estimator_
```

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
[CV 1/5; 1/10] START lasso__C=0.05, tfidf__use_idf=Tru
e.....................
[CV 1/5; 1/10] END .......lasso__C=0.05, tfidf__use_idf=True; total time=
2.6s
[CV 2/5; 1/10] START lasso__C=0.05, tfidf__use_idf=Tru
e.....................
[CV 2/5; 1/10] END .......lasso__C=0.05, tfidf__use_idf=True; total time=
2.1s
[CV 3/5; 1/10] START lasso__C=0.05, tfidf__use_idf=Tru
e.....................
[CV 3/5; 1/10] END .......lasso__C=0.05, tfidf__use_idf=True; total time=
2.2s
[CV 4/5; 1/10] START lasso__C=0.05, tfidf__use_idf=Tru
e.....................
[CV 4/5; 1/10] END .......lasso__C=0.05, tfidf__use_idf=True; total time=
2.2s
[CV 5/5; 1/10] START lasso__C=0.05, tfidf__use_idf=Tru
e.....................
[CV 5/5; 1/10] END .......lasso__C=0.05, tfidf__use_idf=True; total time=
2.1s
[CV 1/5; 2/10] START lasso__C=0.05, tfidf__use_idf=Fals
e.....................
[CV 1/5; 2/10] END ......lasso__C=0.05, tfidf__use_idf=False; total time=
2.1s
[CV 2/5; 2/10] START lasso__C=0.05, tfidf__use_idf=Fals
e.....................
[CV 2/5; 2/10] END ......lasso__C=0.05, tfidf__use_idf=False; total time=
2.1s
[CV 3/5; 2/10] START lasso__C=0.05, tfidf__use_idf=Fals
e.....................
[CV 3/5; 2/10] END ......lasso__C=0.05, tfidf__use_idf=False; total time=
2.1s
[CV 4/5; 2/10] START lasso__C=0.05, tfidf__use_idf=Fals
e.....................
[CV 4/5; 2/10] END ......lasso__C=0.05, tfidf__use_idf=False; total time=
2.1s
[CV 5/5; 2/10] START lasso__C=0.05, tfidf__use_idf=Fals
e.....................
[CV 5/5; 2/10] END ......lasso__C=0.05, tfidf__use_idf=False; total time=
2.4s
[CV 1/5; 3/10] START lasso__C=0.1, tfidf__use_idf=Tru
e.....................
[CV 1/5; 3/10] END ........lasso__C=0.1, tfidf__use_idf=True; total time=
2.1s
[CV 2/5; 3/10] START lasso__C=0.1, tfidf__use_idf=Tru
e.....................
[CV 2/5; 3/10] END ........lasso__C=0.1, tfidf__use_idf=True; total time=
2.1s
[CV 3/5; 3/10] START lasso__C=0.1, tfidf__use_idf=Tru
e.....................
[CV 3/5; 3/10] END ........lasso__C=0.1, tfidf__use_idf=True; total time=
2.1s
[CV 4/5; 3/10] START lasso__C=0.1, tfidf__use_idf=Tru
```

```
e.........................
[CV 4/5; 3/10] END ........lasso__C=0.1, tfidf__use_idf=True; total time=
2.1s
[CV 5/5; 3/10] START lasso__C=0.1, tfidf__use_idf=Tru
e.........................
[CV 5/5; 3/10] END ........lasso__C=0.1, tfidf__use_idf=True; total time=
2.1s
[CV 1/5; 4/10] START lasso__C=0.1, tfidf__use_idf=Fals
e.........................
[CV 1/5; 4/10] END .......lasso__C=0.1, tfidf__use_idf=False; total time=
2.1s
[CV 2/5; 4/10] START lasso__C=0.1, tfidf__use_idf=Fals
e.........................
[CV 2/5; 4/10] END .......lasso__C=0.1, tfidf__use_idf=False; total time=
2.1s
[CV 3/5; 4/10] START lasso__C=0.1, tfidf__use_idf=Fals
e.........................
[CV 3/5; 4/10] END .......lasso__C=0.1, tfidf__use_idf=False; total time=
2.1s
[CV 4/5; 4/10] START lasso__C=0.1, tfidf__use_idf=Fals
e.........................
[CV 4/5; 4/10] END .......lasso__C=0.1, tfidf__use_idf=False; total time=
2.1s
[CV 5/5; 4/10] START lasso__C=0.1, tfidf__use_idf=Fals
e.........................
[CV 5/5; 4/10] END .......lasso__C=0.1, tfidf__use_idf=False; total time=
2.1s
[CV 1/5; 5/10] START lasso__C=0.5, tfidf__use_idf=Tru
e.........................
[CV 1/5; 5/10] END ........lasso__C=0.5, tfidf__use_idf=True; total time=
2.1s
[CV 2/5; 5/10] START lasso__C=0.5, tfidf__use_idf=Tru
e.........................
[CV 2/5; 5/10] END ........lasso__C=0.5, tfidf__use_idf=True; total time=
2.1s
[CV 3/5; 5/10] START lasso__C=0.5, tfidf__use_idf=Tru
e.........................
[CV 3/5; 5/10] END ........lasso__C=0.5, tfidf__use_idf=True; total time=
2.1s
[CV 4/5; 5/10] START lasso__C=0.5, tfidf__use_idf=Tru
e.........................
[CV 4/5; 5/10] END ........lasso__C=0.5, tfidf__use_idf=True; total time=
2.1s
[CV 5/5; 5/10] START lasso__C=0.5, tfidf__use_idf=Tru
e.........................
[CV 5/5; 5/10] END ........lasso__C=0.5, tfidf__use_idf=True; total time=
2.1s
[CV 1/5; 6/10] START lasso__C=0.5, tfidf__use_idf=Fals
e.........................
[CV 1/5; 6/10] END .......lasso__C=0.5, tfidf__use_idf=False; total time=
2.1s
[CV 2/5; 6/10] START lasso__C=0.5, tfidf__use_idf=Fals
e.........................
[CV 2/5; 6/10] END .......lasso__C=0.5, tfidf__use_idf=False; total time=
2.3s
[CV 3/5; 6/10] START lasso__C=0.5, tfidf__use_idf=Fals
e.........................
[CV 3/5; 6/10] END .......lasso__C=0.5, tfidf__use_idf=False; total time=
2.1s
[CV 4/5; 6/10] START lasso__C=0.5, tfidf__use_idf=Fals
e.........................
```

```
[CV 4/5; 6/10] END .......lasso__C=0.5, tfidf__use_idf=False; total time=
2.1s
[CV 5/5; 6/10] START lasso__C=0.5, tfidf__use_idf=Fals
e.........................
[CV 5/5; 6/10] END .......lasso__C=0.5, tfidf__use_idf=False; total time=
2.1s
[CV 1/5; 7/10] START lasso__C=1, tfidf__use_idf=Tru
e.............................
[CV 1/5; 7/10] END ..........lasso__C=1, tfidf__use_idf=True; total time=
2.1s
[CV 2/5; 7/10] START lasso__C=1, tfidf__use_idf=Tru
e.............................
[CV 2/5; 7/10] END ..........lasso__C=1, tfidf__use_idf=True; total time=
2.1s
[CV 3/5; 7/10] START lasso__C=1, tfidf__use_idf=Tru
e.............................
[CV 3/5; 7/10] END ..........lasso__C=1, tfidf__use_idf=True; total time=
2.2s
[CV 4/5; 7/10] START lasso__C=1, tfidf__use_idf=Tru
e.............................
[CV 4/5; 7/10] END ..........lasso__C=1, tfidf__use_idf=True; total time=
2.2s
[CV 5/5; 7/10] START lasso__C=1, tfidf__use_idf=Tru
e.............................
[CV 5/5; 7/10] END ..........lasso__C=1, tfidf__use_idf=True; total time=
2.1s
[CV 1/5; 8/10] START lasso__C=1, tfidf__use_idf=Fals
e.............................
[CV 1/5; 8/10] END .........lasso__C=1, tfidf__use_idf=False; total time=
2.1s
[CV 2/5; 8/10] START lasso__C=1, tfidf__use_idf=Fals
e.............................
[CV 2/5; 8/10] END .........lasso__C=1, tfidf__use_idf=False; total time=
2.1s
[CV 3/5; 8/10] START lasso__C=1, tfidf__use_idf=Fals
e.............................
[CV 3/5; 8/10] END .........lasso__C=1, tfidf__use_idf=False; total time=
2.2s
[CV 4/5; 8/10] START lasso__C=1, tfidf__use_idf=Fals
e.............................
[CV 4/5; 8/10] END .........lasso__C=1, tfidf__use_idf=False; total time=
2.1s
[CV 5/5; 8/10] START lasso__C=1, tfidf__use_idf=Fals
e.............................
[CV 5/5; 8/10] END .........lasso__C=1, tfidf__use_idf=False; total time=
2.2s
[CV 1/5; 9/10] START lasso__C=5, tfidf__use_idf=Tru
e.............................
[CV 1/5; 9/10] END ..........lasso__C=5, tfidf__use_idf=True; total time=
3.2s
[CV 2/5; 9/10] START lasso__C=5, tfidf__use_idf=Tru
e.............................
[CV 2/5; 9/10] END ..........lasso__C=5, tfidf__use_idf=True; total time=
2.9s
[CV 3/5; 9/10] START lasso__C=5, tfidf__use_idf=Tru
e.............................
[CV 3/5; 9/10] END ..........lasso__C=5, tfidf__use_idf=True; total time=
3.2s
[CV 4/5; 9/10] START lasso__C=5, tfidf__use_idf=Tru
e.............................
[CV 4/5; 9/10] END ..........lasso__C=5, tfidf__use_idf=True; total time=
```

```
2.9s
[CV 5/5; 9/10] START lasso__C=5, tfidf__use_idf=Tru
e.........................
[CV 5/5; 9/10] END ..........lasso__C=5, tfidf__use_idf=True; total time=
3.0s
[CV 1/5; 10/10] START lasso__C=5, tfidf__use_idf=Fals
e.........................
[CV 1/5; 10/10] END ........lasso__C=5, tfidf__use_idf=False; total time=
3.1s
[CV 2/5; 10/10] START lasso__C=5, tfidf__use_idf=Fals
e.........................
[CV 2/5; 10/10] END ........lasso__C=5, tfidf__use_idf=False; total time=
3.0s
[CV 3/5; 10/10] START lasso__C=5, tfidf__use_idf=Fals
e.........................
[CV 3/5; 10/10] END ........lasso__C=5, tfidf__use_idf=False; total time=
3.3s
[CV 4/5; 10/10] START lasso__C=5, tfidf__use_idf=Fals
e.........................
[CV 4/5; 10/10] END ........lasso__C=5, tfidf__use_idf=False; total time=
3.1s
[CV 5/5; 10/10] START lasso__C=5, tfidf__use_idf=Fals
e.........................
[CV 5/5; 10/10] END ........lasso__C=5, tfidf__use_idf=False; total time=
3.1s
```

In [16]:

```
#Finding the best performing model and saving it
bestlasso = lasso_result.best_estimator_

print('Here we se the parameters of the best Lasso model:',lasso_result.best_params_,
      '\nAvg. accuracy score of the best performing model:',lasso_result.best_score_)
```

```
Here we se the parameters of the best Lasso model: {'lasso__C': 5, 'tfidf__u
se_idf': False}
Avg. accuracy score of the best performing model: 0.9581516039849373
```

In [17]:

```
#Investigate the difference between using plain frequencies and tf-idf

#Saving the results of all parameters in a dataframe
lasso_result_df = pd.DataFrame(lasso_result.cv_results_)

#Locating and printing the accuracy results of the two vectorizers with the optimal C param
print('Result of best model with tf-idf vectorizing:\t',
      lasso_result_df.loc[(lasso_result_df.param_tfidf__use_idf == True)
                          & (lasso_result_df.param_lasso__C == 5)]['mean_test_score'].value

print('Result of best model with count vectorizing:\t',
      lasso_result_df.loc[(lasso_result_df.param_tfidf__use_idf == False)
                   & (lasso_result_df.param_lasso__C == 5)]['mean_test_score'].values)


#Here we see that the tf-idf vectorizing performs better than the count vectorizer.
```

```
Result of best model with tf-idf vectorizing:    [0.9572252]
Result of best model with count vectorizing:     [0.9581516]
```

In [18]:

```
lasso_result_df
```

Out[18]:

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_lasso__C	param_tfidf__
0	1.899923	0.157432	0.351921	0.016565	0.05	
1	1.796812	0.113236	0.360305	0.022778	0.05	
2	1.755026	0.027815	0.332348	0.028881	0.1	
3	1.732984	0.021116	0.353035	0.017924	0.1	
4	1.753875	0.026370	0.358705	0.017523	0.5	
5	1.807679	0.088731	0.332514	0.028496	0.5	
6	1.794865	0.041777	0.356022	0.013506	1	
7	1.815354	0.061064	0.345132	0.016560	1	
8	2.664473	0.138621	0.350346	0.018049	5	
9	2.769078	0.105106	0.344397	0.014190	5	

In [ ]:

```

```

In [19]:

```
#drop words from proc_text_all which are in our classifier lists
unlabeled['tokens_without_class'] = [[word for word in words
                                      if word not in human + critism]
                                     for words in unlabeled["tokens"]]

unlabeled['text_w_class'] = unlabeled['tokens_without_class'].apply(
                      lambda x: " ".join( x))
#unlabeled['tokens_without_class']  = [word for word in unlabeled['tokens'] if word not in
#result = ' '.join(resultwords)

#print(result)
```

In [20]:

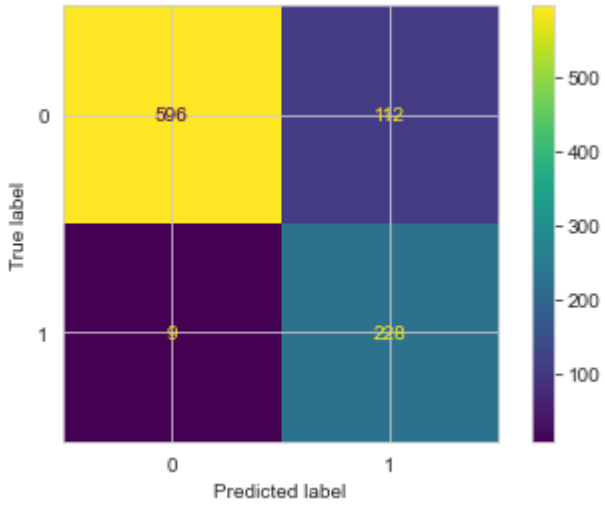
```
print(len(unlabeled.tokens[1]))
print(len(unlabeled.tokens_without_class[1]))
#unlabeled.head()
```

```
47
41
```

In [38]:

```
#Splitting unlabeled features into X and y

Xnew = unlabeled['proc_text_all']
#Xnew = unlabeled['text_w_class']
ynew = unlabeled['class_type'].values


print('Accuracy and confusion matrix for best Lasso')

#Fitting the best model with the labeled dataset
bestlasso.fit(X, y)

#Getting the accuracy score
print(bestlasso.score(Xnew, ynew))

#Confusion matrix Lasso
plot_confusion_matrix(bestlasso, Xnew, ynew)
plt.show()
#plt.savefig('Accuracy and confusion matrix for best Lasso.png')
```

```
Accuracy and confusion matrix for best Lasso
0.8719576719576719
```



In [22]:

```
#Getting predicted values with Lasso
ypred = bestlasso.predict(Xnew)

#Getting classification report for Lasso
print(classification_report(ypred,ynew))
```

```
              precision    recall  f1-score   support

           0       0.84      0.99      0.91       605
           1       0.96      0.67      0.79       340

    accuracy                           0.87       945
   macro avg       0.90      0.83      0.85       945
weighted avg       0.89      0.87      0.87       945
```

In [23]:

```
#Repeating above steps for Lasso

n_classes = 2

fpr = dict()
tpr = dict()
roc_auc = dict()

probs = bestlasso.predict_proba(Xnew)

for i in range(n_classes):
    fpr[i], tpr[i], threshold = roc_curve(ynew, probs[:, i])
    roc_auc[i] = auc(fpr[i], tpr[i])
```

In [24]:

```
human_pred = []
critism_pred = []
for element in probs:
    human_pred.append(element[0])
    critism_pred.append(element[1])
```

In [25]:

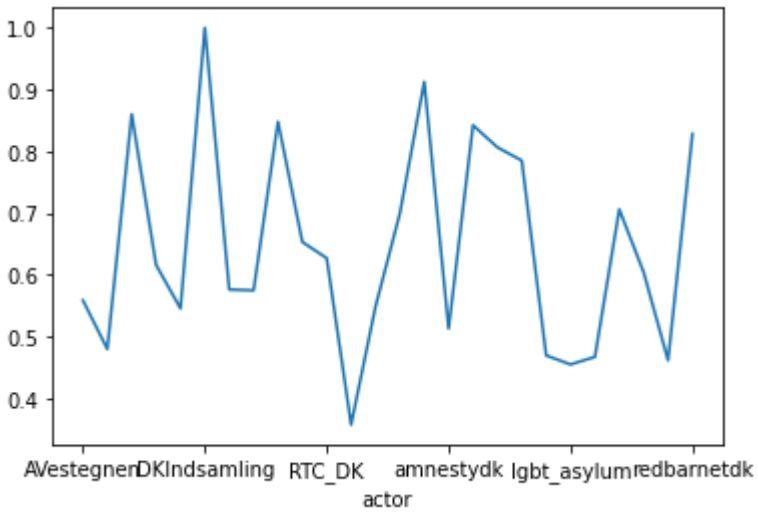
```
unlabeled['human_pred'] = human_pred
unlabeled['critism_pred'] = critism_pred
```

In [26]:

```
unlabeled.groupby('actor')['human_pred'].mean().plot()
```

Out[26]:

```
<AxesSubplot:xlabel='actor'>
```



In [27]:

```
critism_p = unlabeled.groupby('actor')['critism_pred'].mean()
human_p = unlabeled.groupby('actor')['human_pred'].mean()
print("critism predictions:", unlabeled.groupby('actor')['critism_pred'].mean())
print("human predictions:", unlabeled.groupby('actor')['human_pred'].mean())
```

```
critism predictions: actor
AVestegnen          4.409537e-01
ActionAidDK         5.198059e-01
BornsVilkar         1.397510e-01
CARE_Danmark        3.835960e-01
CaritasDanmark      4.540559e-01
DKIndsamling        8.099859e-08
DRC_dk              4.233679e-01
DignityDK           4.248262e-01
MissionEast         1.519270e-01
PlanBornefonden     3.465678e-01
RTC_DK              3.727310e-01
RefWelcome          6.420300e-01
UNDP_Danmark        4.511161e-01
UNICEFDK            3.003533e-01
WFP_DK              8.742074e-02
amnestydk           4.866119e-01
blaakorsdanmark     1.576154e-01
danmissiondk        1.933438e-01
danskrodekors       2.148505e-01
globaltfokus        5.300567e-01
lgbt_asylum         5.446839e-01
menneskeret         5.324441e-01
msf_dk              2.936944e-01
noedhjaelp          3.957100e-01
oxfamibis           5.379452e-01
redbarnetdk         1.717436e-01
Name: critism_pred, dtype: float64
human predictions: actor
AVestegnen          0.559046
ActionAidDK         0.480194
BornsVilkar         0.860249
CARE_Danmark        0.616404
CaritasDanmark      0.545944
DKIndsamling        1.000000
DRC_dk              0.576632
DignityDK           0.575174
MissionEast         0.848073
PlanBornefonden     0.653432
RTC_DK              0.627269
RefWelcome          0.357970
UNDP_Danmark        0.548884
UNICEFDK            0.699647
WFP_DK              0.912579
amnestydk           0.513388
blaakorsdanmark     0.842385
danmissiondk        0.806656
danskrodekors       0.785149
globaltfokus        0.469943
lgbt_asylum         0.455316
menneskeret         0.467556
msf_dk              0.706306
```

```
noedhjaelp          0.604290
oxfamibis           0.462055
redbarnetdk         0.828256
Name: human_pred, dtype: float64
```

In [62]:

```
```

Out[62]:

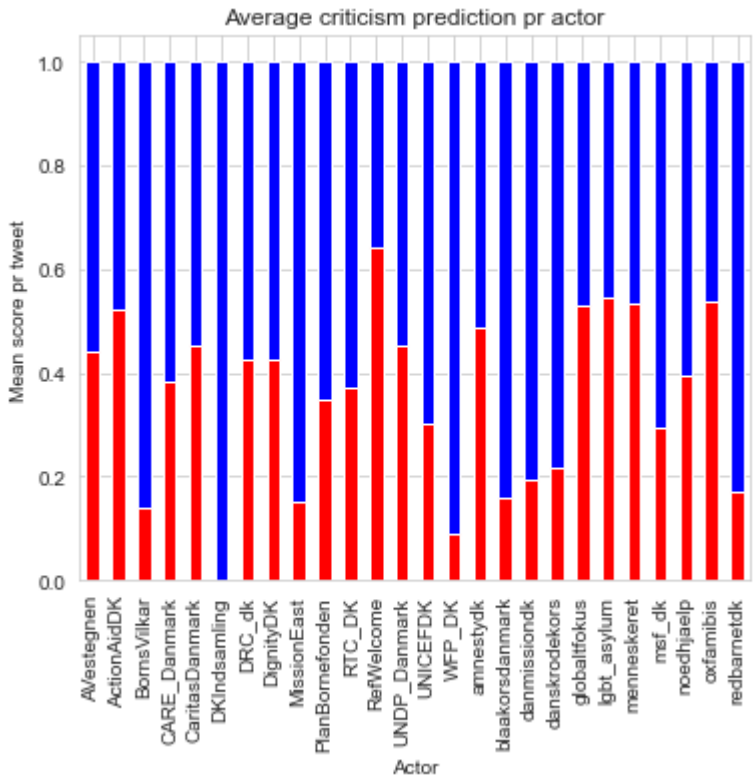
```
0.4409537033272377
```

In [56]:

```
human_p2 = 1 + 0 * critism_p
```

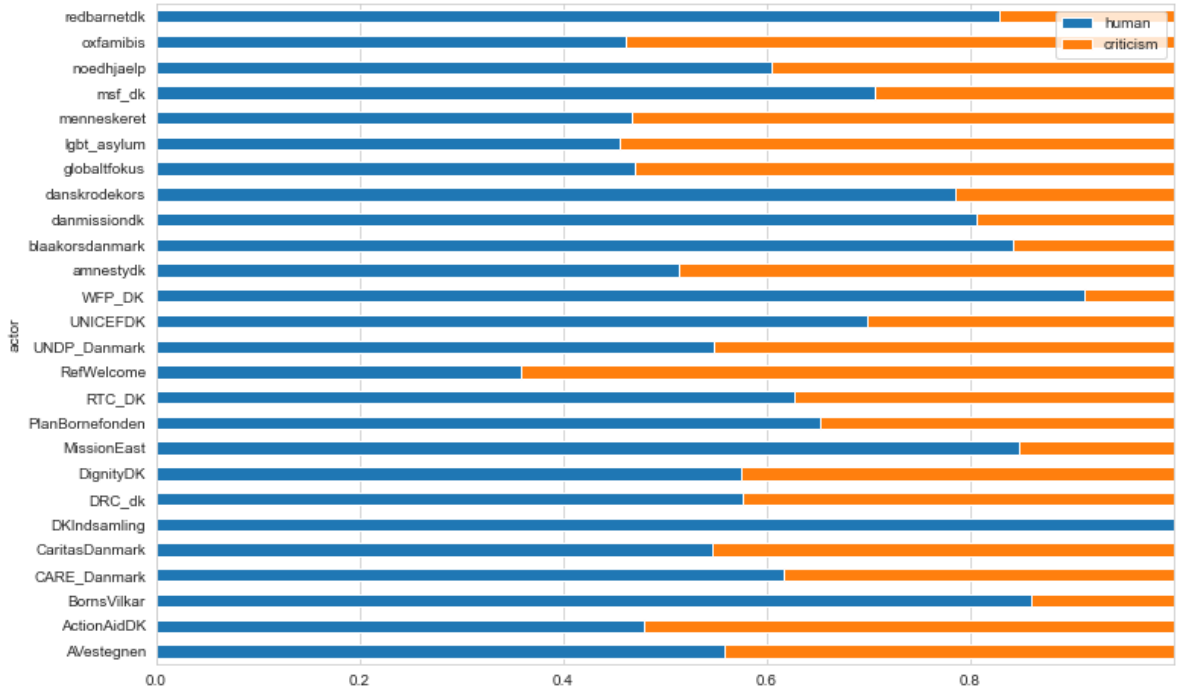
In [59]:

```
human_p2.plot(kind='bar', title='Average criticism prediction pr actor', ylabel='Mean score
        xlabel='Actor', figsize=(6, 5), color="blue")
critism_p.plot(kind='bar', title='Average criticism prediction pr actor', ylabel='Mean scor
        xlabel='Actor', figsize=(6, 5), color="red")

plt.savefig('ave score of critism.png', bbox_inches='tight')
```



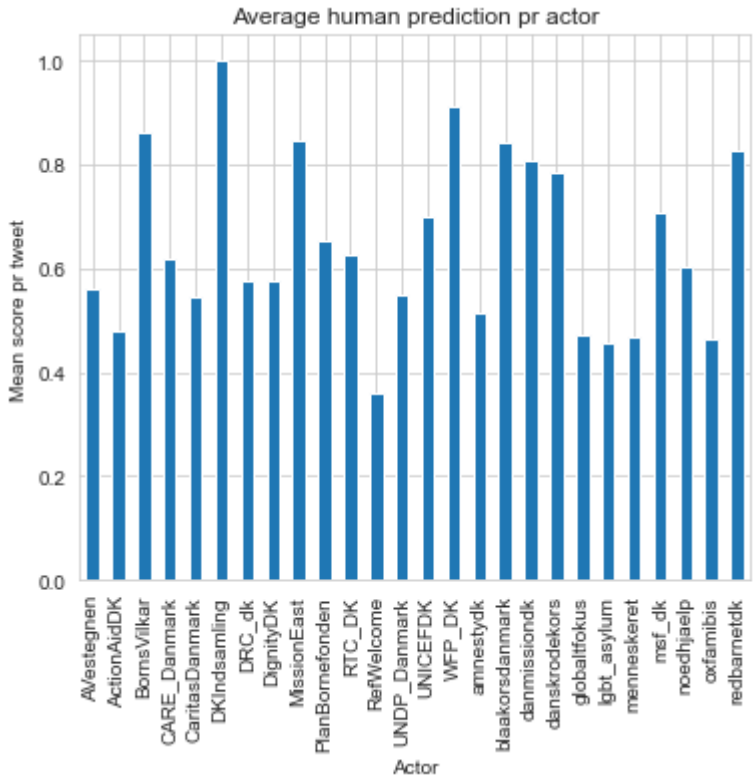
In [71]:

```
df = pd.DataFrame({'human': human_p,
                   'criticism': critism_p})
ax = df.plot.barh(stacked=True, figsize=(12,8))
plt.legend(loc="upper right")
plt.savefig('human_critisism_plot.png', bbox_inches='tight')
```



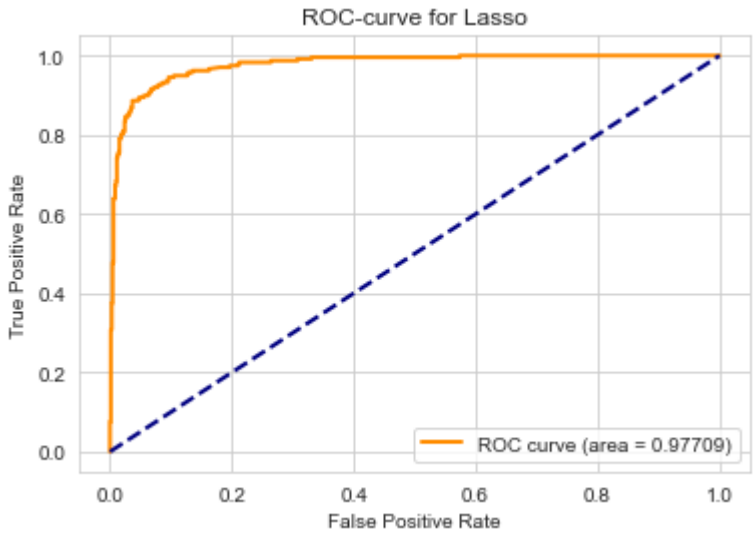
In [43]:

```
human_p.plot(kind='bar', title='Average human prediction pr actor', ylabel='Mean score pr t
         xlabel='Actor', figsize=(6, 5))
plt.savefig('ave score of human.png',bbox_inches='tight')
```



In [41]:

```
#Plotting the ROC-curve and AUC-score for Lasso

plt.figure()

lw = 2

plt.plot(fpr[1], tpr[1], color='darkorange', lw=lw, label='ROC curve (area = %0.5f)' % roc_

plt.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')

plt.title('ROC-curve for Lasso')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')

plt.legend(loc="lower right")

#plt.show()
plt.savefig('ROC-curve for Lasso.png')
```



Coefficient from Lasso model

In [31]:

```
import numpy as np
```

In [32]:

```
#Getting the feature names
feature_names = np.array(bestlasso.named_steps["vect"].get_feature_names())
feature_names.shape
```

Out[32]:

```
(329,)
```

In [33]:

```
#Creating a dataframe with lasso coefficients (similar to above)
lasso_coefs = pd.DataFrame(np.vstack((feature_names, bestlasso.named_steps['lasso'].coef_))
                                    columns = ['feat_name', 'coef'])
#Converting column values from string type to float
lasso_coefs['coef'] = lasso_coefs['coef'].astype(float)

#Sort the features by the absolute value of their coefficient

#Creating a column for absolute values
lasso_coefs["abs_value"] = lasso_coefs["coef"].apply(lambda x: abs(x))

#Creating a column of colors, based on whether the coefficient is positive or negative
#(This step is not necessary - just for nice visualization)
lasso_coefs["colors"] = lasso_coefs["coef"].apply(lambda x: "green" if x > 0 else "red")

#Sorting the dataframe based on the absolute value column
lasso_coefs = lasso_coefs.sort_values("abs_value", ascending=False)
```

In [34]:

```
#Viewing the top 20 most impactful coefficients
lasso_coefs[:20]
```

Out[34]:

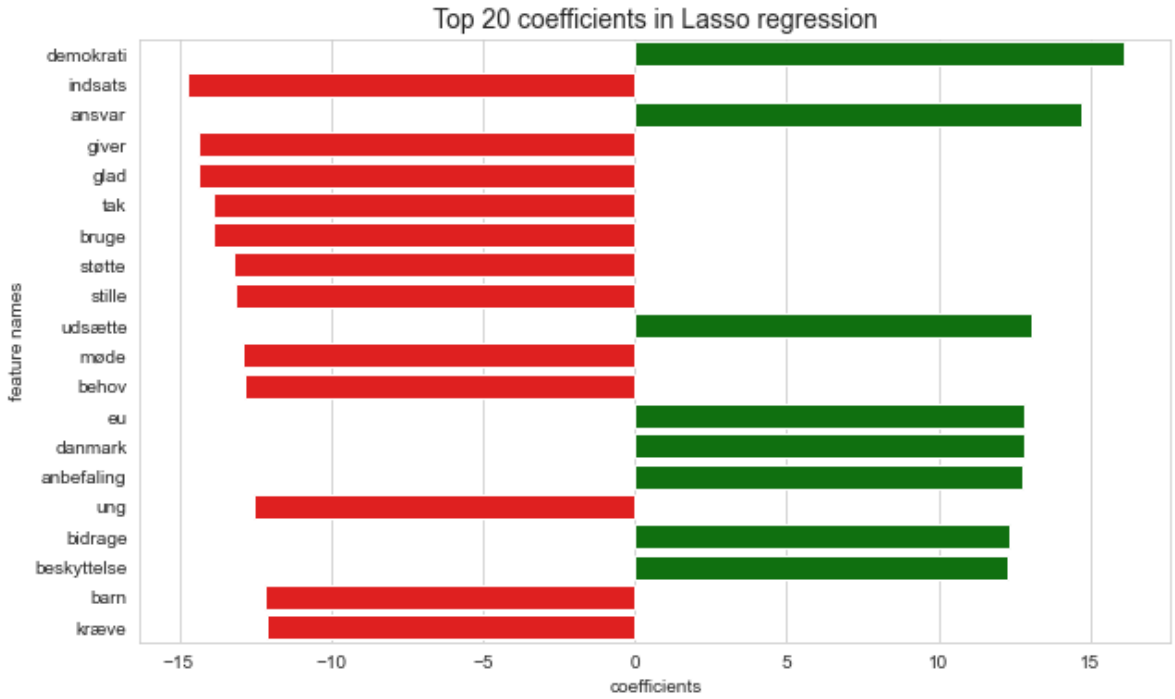
	feat_name	coef	abs_value	colors
56	demokrati	16.103151	16.103151	green
149	indsats	-14.738250	14.738250	red
17	ansvar	14.704586	14.704586	green
118	giver	-14.370944	14.370944	red
119	glad	-14.363528	14.363528	red
287	tak	-13.878446	13.878446	red
37	bruge	-13.848339	13.848339	red
275	støtte	-13.229552	13.229552	red
266	stille	-13.160146	13.160146	red
303	udsætte	13.087177	13.087177	green
206	møde	-12.898409	12.898409	red
27	behov	-12.868054	12.868054	red
80	eu	12.792862	12.792862	green
48	danmark	12.790836	12.790836	green
15	anbefaling	12.764096	12.764096	green
308	ung	-12.566299	12.566299	red
34	bidrage	12.334406	12.334406	green
31	beskyttelse	12.242537	12.242537	green
22	barn	-12.176523	12.176523	red
163	kræve	-12.130359	12.130359	red

In [35]:

```
import seaborn as sns
```

In [48]:

```
sns.set_style('whitegrid')

#Plotting the top 20 most impactful coefficients
plt.figure(figsize = (10,6))

sns.barplot(x = 'coef',y = 'feat_name', data = lasso_coefs[:20], palette = lasso_coefs[:20]
plt.ylabel('feature names')
plt.xlabel('coefficients')
plt.title('Top 20 coefficients in Lasso regression', fontsize = 14);

plt.savefig('top 20 coef in Lasso.png')
```



In [ ]: