Responding to Computational Propaganda

Pedro A. Arroyo, Daejin Kim, Arun Pandian, Di Tong

12/11/2019

Contents

NOTE: Appendices are located after References.

Contribution Statement

This report is the result of a collaboration between the authors. Each author was involved throughout the process and, ultimately, all sections of the report depended on input from all particpants. Even so, we report here what each author feels were their most significant contributions:

1. **Arroyo**: Project design, principally responsible for drafting the Findings section, primarily responsible for editing and drafting of the final report.
2. **Kim**: Identifying relevant literature, principally responsible for drafting the Data Collection section ('Building a Corpus') and the Conclusion and jointly responsible for drafting the Introdcution, located the report (McGann 2019) that made our particular analysis..
3. **Pandian**: Initial question design, code design (including text preprocessing and cleaning), principally responsible for drafting the Preprocessing section and jointly responsible for drafting the Introduction.
4. **Tong**: Methodological design, code design (including dynamic topic model, tuning topic number, and visualization), principally responsible for drafting the Methods section.

In addition, all members of the group participated in the iterative process of analyzing model output as well as the life-affirming process of pulling reports and converting pdfs to .txt format, followed by manual cleaning.

Disclaimers

This report is rendered as a single-space document. We encountered problems in mixing both single and double space formats; we opted for single-space because doublespace produced an inelegant and unwieldy document.

1 Introduction

With the move of public political discourse from traditional media to relatively democratized digital platforms, the United States is increasingly coming into contact with computational propaganda - i.e. the use of algorithms and automation to disseminate false information or to manipulate user activity on social media (Wolley and Howard 2017). Researchers, politicians, and policy experts have recognized the emergence of these developments as a threat to democratic institutions. Concerns have grown over the corrosive effects of strategic competitors using distorted, misleading, or fabricated information to affect political behavior. For example, the 2016 election saw pro-Trump campaigns leverage computational techniques to generate pro-Trump twitter accounts. By election day, the number of pro-Trump twitter accounts outnumbered pro-Clinton accounts by a factor of 5 to 1 (Kollanyi, Howard, and Wolley 2016). In addition, the Mueller investigation found evidence of anti-Trump protests in the United Statesorganized by foreign actors (Weiss 2018). We can see that this is a complicated political landscape where actors, their intentions, and beneficiaries do not seem to fit into a simple narrative; and yet, computational disinformation techniques threaten the democratic process on two fronts: they insert false information into the political consciousness and they undermine the franchise of individual voters.

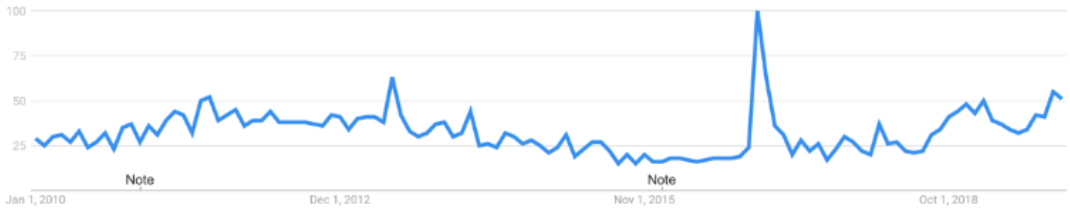


Figure 1: Google Web Search Frequency of "disinformation", 2010-2019

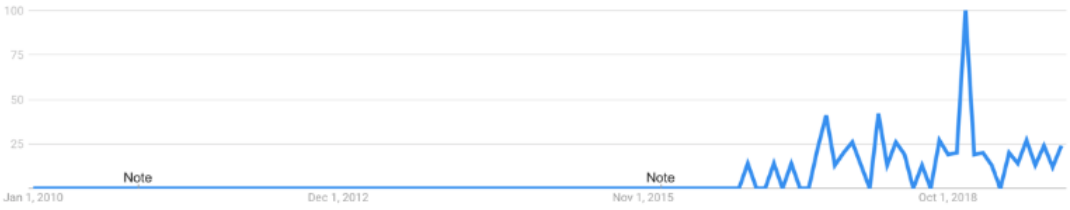


Figure 2: Google Web Search Frequency of "Computational Propaganda", 2010-2019

This issue has certainly caught the attention of voter (see **Figures 1 and 2**). It has also garnered a response from the world of think tanks in the form of reports and working papers attempting to frame computational propaganda and/or outline a counter-strategy. This conversation is necessarily narrow in terms of participants, including only a limited slice of elite discourse. That slice, however, is important in so far as it includes the decision-makers that will most heavily influence an organized response.

To gain a better understanding of how the most prominent think tanks approach and react to computational propaganda, we analyzed their published reports. We have used temporal topic modeling to classify text data. Specifically we used Dynamic Topic Modelling, which we believe can help us identify semantically cohesive topics in the corpus and gain insights into how computational disinformation is conceptualized in this discourse. Additionally, it can also help us investigate how topics have evolved in time. Our goal in deriving these topics is not to definitively identify the x-number of topics that comprise the corpus, but rather to identify coherent and stable dimensions of meaning that can then form the basis for future study.

2 Building a Corpus

To create a corpus, we first selected a sample of think tanks that can capture dominant expert views. To do so, we relied on the list of the top twenty U.S. think tanks as reported in "2018 Global Go To Think Tank Index Report" (McGann 2019). The index report examines more than eight thousands think tanks around the world, which includes institutions across the ideological spectrum. It measures the influence level of each think tank by taking both qualitative (e.g. leadership, reputation, academic performance) and quantitative (e.g. the number of publications and policy proposals) criteria (See 31-34 pages of the report for the complete list). By choosing the top twenty U.S. think tanks from this report, we hope to capture a conservative, though limited, approximation of the discourse of policy architects.

Reports were collected from the selected think tanks using the following method. For think tanks that tag their articles, we included reports with the following tags: "online disinformation", "computational propaganda", "influence campaign", "disinformation", and "propaganda." Using the last three tags, we checked the reports to make sure that they discussed online disinformation/propaganda, not off-line disinformation. Reports tagged with "online disinformation" and "computational propaganda" could be included without further investigation because the tags made it very clear that they were on digital disinformation. With sources that did not tag their reports, we used the aforementioned keywords as search terms and used our judgment to decide inclusion based on titles and tables of contents. Although this searching process does not ensure that we collected all the relevant reports, it at least guarantees that the selected reports deal with our topic of interest. This produces a selective sample of expert discourse on this issue.

Year	**Number of Reports**
2016	3
2017	14
2018	34
2019	25
Total	**76**

Table 1. Meta-information by year

Rank	**Think Tank**	# of Reports
1	Brookings Institution	13
2	Center for Strategic and International Studies	3
3	Carnegie Endowment for International Peace	1
4	Heritage Foundation	1
6	RAND Corporation	11
8	Center for American Progress	16
10	Atlantic Council	8
11	Council on Foreign Relations	6
14	Belfer Institute	13
15	Hudson Institute	2
16	American Enterprise Institute	1
19	Stimson Center	1
Total		**12**

Table 2. Think tanks included in the corpus by the ranking and the number of reports.

2.1 Preprocessing

After collecting reports, most in the form of PDFs, we used PDFMiner (Shinyama 2019), to convert these reports to text format. We then skimmed through the texts manually to find words that constitute domain-specific stopwords (**See Appendix A**). This is followed by an initial round of cleaning[1], in which we:

1. tokenized the text using spaCy (Honnibal and Johnson 2015),
2. removed English stopwords using spaCy,
3. removed domain-specific stopwords using spaCy,
4. removed numbers and types with numbers,
5. removed types with non-english characters,
6. removed types containing 'www.' or 'http:',
7. removed types that were punctuation, and
8. standardized the text by lowercasing all types and stemming (Lopwer and Bird 2002) the text.

At this point, as specified in the Methods section, we fit a number of models and found an appropriate value for k. We inspected these results and then did another round of cleaning, adding to our dictionary of domain-specific stopwords as well as joining and transforming

[1] *Removal of words with numbers and non-english characters is done to eliminate artifacts of the pdf to text conversion.*

words that are properly understood as constituting one term (**See Appendix B**). We created a domain-specific stopword list of 36 types, and transformed 8 types. We used the 'kwic' (Benoit et al. 2018) function in the R package *quanteda* to check the co-occurrences of types to make sure stopwords and new types were created appropriately.

3 Methods

Our analyses rely on topic modeling, a Bayesian generative method that serves to classify text data (Grimmer and Stewart 2013). As an unsupervised machine learning technique, topic modeling has the potential to either confirm existing theories or discover unknown categories and patterns not immediately apparent to human readers (Blei, Ng, and Jordan 2003; Evans and Aceves 2016; Nelson et al. 2018). Compared with a more basic form of text-classification method, Latent Semantic Analysis (LSA), topic modeling is better at discovering "semantically cohesive topics and their combination across document collections (Evans and Aceves 2016)." Latent Dirichlet Allocation (LDA) is the most prevalently-used algorithm for topic modeling. Assuming that each document contains a mixture of topics, some of which occur throughout the entire corpus, LDA uncovers hidden thematic structures both in the corpus and in individual documents. Specifically, it can identify (i) the topics defined by a group of words and their weights for each topic and (ii) the distribution of topics in the corpus as well as in an individual document (see **Figure 3**).

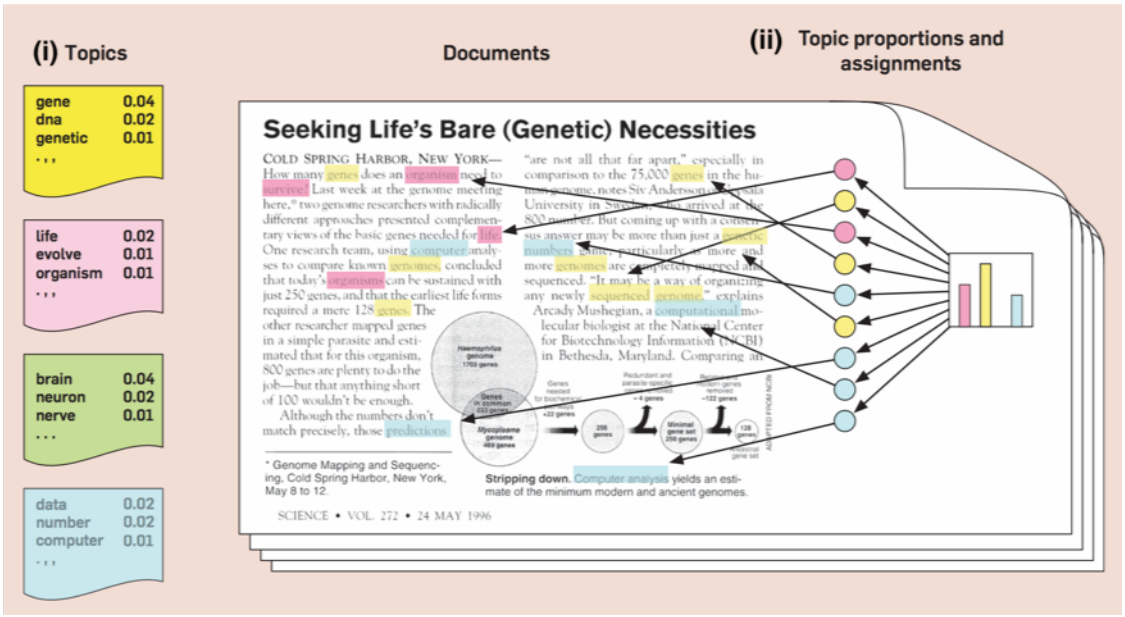


Figure 3: From Blei 2002

Specifically, we use Dynamic Topic Modeling (DTM), an extension of LDA that handles sequential documents (Blei and Lafferty 2016). Based on LDA, DTM allows topic contents and prevalence to evolve over time. The overall topic distribution and the term distribution for each topic differ depending on the time slice. We fit a DTM with four time periods

(year 2016, 2017, 2018, and 2019) to first, establish the shared dominant topics for all texts in our corpus to get a general sense of the major concerns regarding disinformation and computational campaigns in the United States; and second, compare the evolving key word composition of each topic and topic distribution across the four periods to trace and analyze the evolution of (latent) topics over the 4 years following the 2016 presidential election.

After examining the topic coherence metric and the weighted word lists for six models with different topic numbers ranging from 4-10, we choose the model with 4 topics. As is shown in **Figure 4**, the 4-topics model has the highest average topic conference score, indicating that it outperforms the other models. Through measuring the degree of semantic similarity between high weight words for a given topic, topic coherence metric assist us to distinguish between semantically interpretable topics and artifacts of statistical inference (Mimno et al. 2011). We also check the key word composition of all models and find that the 4-topic model produces the most substantively interpretable and non-repetitive combination of topics, which is considered to be the most important criterion to determine the number of topics in social science research (DiMaggio 2015; Nelson 2017).

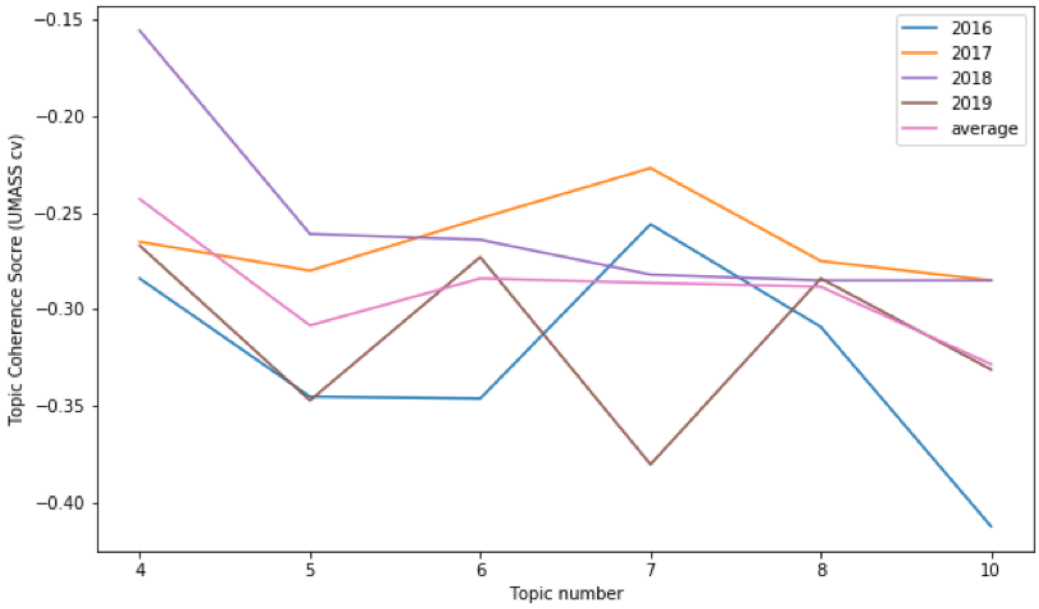


Figure 4: Topic model performance by number of topics.

4 Findings

For purposes of interpretation and analysis, we gave a label to each of the four topics in our model. We believe these labels capture the main thematic content of the keys that load on that topic and, therefore, establish the conceptual coherence of that topic. For each of these topics, we also identify the terms that we found to give us the most interpretative leverage.

These are not always the terms that are most strongly associated with that topic, but rather the terms that, when compared across the term-distribution across topics, makes this topic distinct and, therefore, we believe characterize it. These labels are not definitive, but we believe them to be informative.

We first present and describe the topics in a broad sense, followed by a discussion of changes in topic composition over time. Finally, we discuss changes in relative topic prominence over the time investigated in this study. (We report topics and keys in **Appendix C**.)

4.1 Topics

The first topics is labeled **Security/Russian Threat** and is primarily characterized by the following terms: *Russia, state, attack, foreign, secur(ity), hack,* and *election*. This topic seems to capture concerns about the security threat posed by Russia and emphasizes recent disruptive Russian activity as it pertains to the ongoing conflict in Syria, the illegal annexation of Crimea, and interference in Western elections.

The second topic is labeled **Fake News/Social Media** and is primarily characterized by the following terms: *news, disinform(ation), inform, fake (news), media, social media, online, false, truth,* and *fact*. This topic addresses the increasing phenomenon of active disinformation in the news media as a persistent feature of public discourse and reflects the impression that social media is central to this process.

Key	*Note*
'Manafort'	Paul Manafort was the campaign manager for Donald J. Trump's presidential campaign. He is currently serving a prison sentence in connection to actions associated with that campaign.
'madcom'	MADCOM refers to "the integration of AI systems into machine-driven communications tools for use in computational propaganda". (Atlantic Council), n.d.)

The third topic is labeled **Presidential Campaign** and is primarily characterized by the following terms: *Trump, Russia, Campaign, Putin, Manafort, (new) york, special (counsel),* and *investig(ation)*. This topic captures discussion of the 2016 Presidential election in the United States in the context of Russian interference, as well as the investigations that followed it. Both the Justice Department's Special Investigator probe as well as the investigations conducted by the office of the United States Attorney for the Southern District of New York; an office with broad jurisdiction and resources.

The fourth and final topic is labeled **Digital disinformation campaigns and response** and is characterized by the following terms: *inform, madcom, ai, technology, online,*

machin(e), cyber, internet, data, threat, attack, protect, and *vulner(able)*. This topic captures the conversation around the more technical aspects of disinformation.

4.1.1 Validation

To ensure we did not overfit our model by cleaning data to fit our interpretation, we performed a modest internal validation test. Since each individual document is treated as a 'bag-of-words' by the DTM method, we randomly sampled, in a reproducible manner, 75 percent of the tokens in each of the document. We then fit our model to this data subset using the same hyperparameters.

This approach changed some of the loadings of keys, but produced topics in keeping with the topics the model of our full corpus produced. We concluded from this that it is unlikely that we overfit our model through text cleaning. (We report topics and keys for the validation subset in **Appendix D**.)

4.2 Changes over Time

In this model, every word is associated with every topic; differences in topics are identified by the different weights that words receive within those topics. Strictly speaking, it would be incorrect to say that a term is associated with one term in topic A and not associated with topic B. However, we report the top 20 terms that load on each topic and therefore base our analysis of change over time on two factors: **(i)** which terms load within the top 20 for that topic, and **(ii)** what is the relative prominence of those terms within a topic?

We identify two changes that occured within the composition of topics over the time period captured by our analysis.

4.2.1 Diminishing Centrality of 2016-Specific Events and Investigations

Topic 3, **Presidential Campaign**, addresses both the Trump presidential campaign and the investigations that followed it. This topic becomes more heterogeneous over time. The 2016 loadings are tightly-focused on the 2016 election and the immediate investigation of the Trump campaign. Over time, we see that addition of the terms europe, arab, and japan; possibly reflecting a shift in the framing of campaign interference from an event-centered analysis to a process-centered analysis; that is, not just an analysis of the 2016 campaign, but also a focus on broader structures and actors.

Another important change in the composition of this topic is the appearance of terms associated with the National Rifle Association: *nra, nratv*, and *gun*. (The NRA was found to have been involved in efforts to coordinate illegal campaign spending in support of the Trump campaign, with possible ties to foreign money from Russia.)

These changes support the inference that this topic was highly sensitive to current events and that the changes in the topic over time capture this tendency.

4.2.2 Broadening Context

We identify an increase in keys associated with the democratic process. We see this most strikingly in topic 2 (**Fake News/Social Media**) with the arrival of the following terms in 2018 and continuing into 2019: *democrat, democraci(es), parti(es), institut(ions, govern(ment), state*, and *polit(ics)*. This may be related to the US electoral cycle, which bookends our period with presidential elections.

Another change relates to which global actors are prominent in different periods. Though Russia remains a prominent actor throughout the time period of this study, other global actors wax and wane in importance.

We mentioned earlier the shift away from Ukraine and Syria and towards the broader framing of Europe. A potentially more significant shift, from a strategic perspective, is the emergence of China as an actor in 2019 in topic 1 (**Security/Russian Threat**), reflected in the appearance of *China* and *Taiwan* as keys.

4.2.3 Changes in Topic Prominence

We identify three changes in topic prominence over our study period.

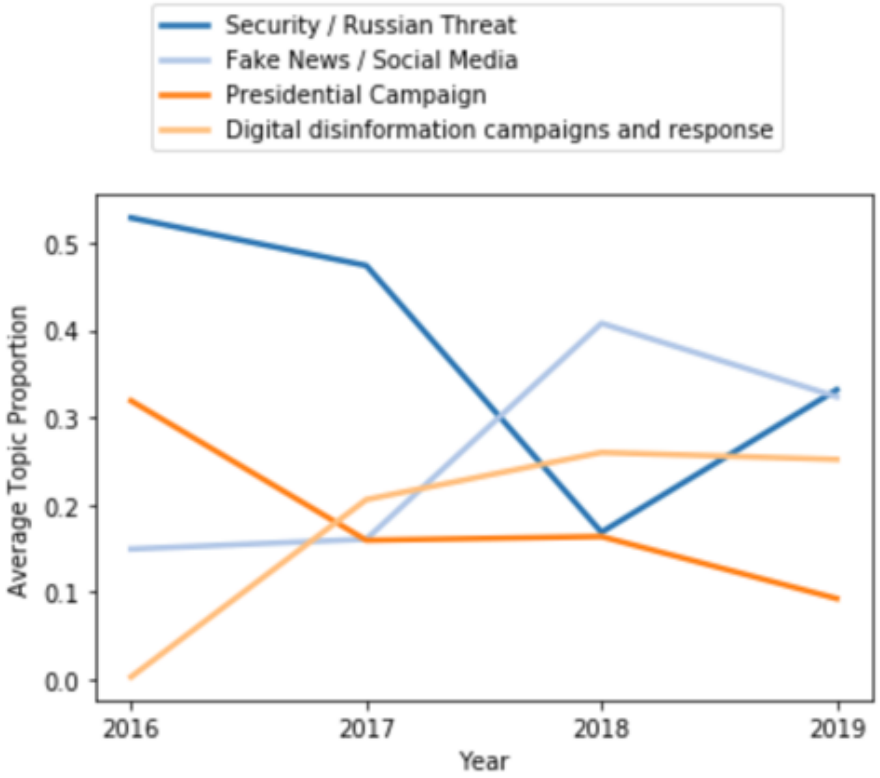


Figure 5: Evolution in Topic Prevalence.

First, the 2016 presidential campaign becomes less prominent over time - this trend is compatible with our earlier inference that discourse shifted from a focus on events to a focus on strategic concerns. This interpretation is strengthened by the in-topic change in **Presidential Campaign** discussed earlier. (It is interesting to note that this is an observation that would most likely be missed with a different analytical approach. This is because both the quantity and composition of texts is changing over time, meaning that, even as the relative frequency of **Presidential Campaign** as a topic is diminishing, the total occurrence of it is increasing.)

Second, topic four (**Digital disinformation campaigns and response**) emerges as an important topic and remains one. In 2016, topic 4 was, effectively, not a theme in the corpus. By 2017, it is the second most prominent topic.

Finally, there appears to be an interesting interplay between topics 1 and 2 (**Security/Russian Threat** and **Fake News/Social Media**, respectively.) From 2016 to 2018, **Security/Russian Threat** diminishes in importance while **Fake News/Social Media** increases in importance. We interpret this as a shift towards a more detail and process oriented conversation.

In 2019, both topics converge. We interpret this as a renewed focus on the digital threat, possibly in light of the upcoming 2020 presidential election. One interpretation of this trend is that the security threat was originally viewed as the central issue pertaining to digital propaganda; but, by the end of the period, fake news and social media is recognized to be equally salient. We also note that, as we saw earlier, **Security/Russian Threat** now includes China as an important actor, representing another way in which the framing of digital propaganda is more diverse at the end of the study period when compared to the beginning of the period.

If we treat the security threat as the most (substantively) important aspect of discourse in these texts, we can posit that American policy makers came to talk about that threat differently in 2019 when compared to 2016 in two ways: **(i)** they recognized the centrality of fake news and social media, and **(ii)** they expanded their understanding of relevant actors to include China.

4.2.4 Validation

Here, we report the evolution in topic prevalence in the subset corpus described above:

The evolution of topics is different in the validation subset than in the full corpus, which implies that our findings on topic prevalence are less robust than our findings on topic composition. Comparing the patterns, it looks like there is a discrepency in whether some documents are associated with **Security / Russia Threat** or **Fake News / Social Media**, indicating that the distinction between the two topics might be *porous*. It is difficult to say more without further analysis.

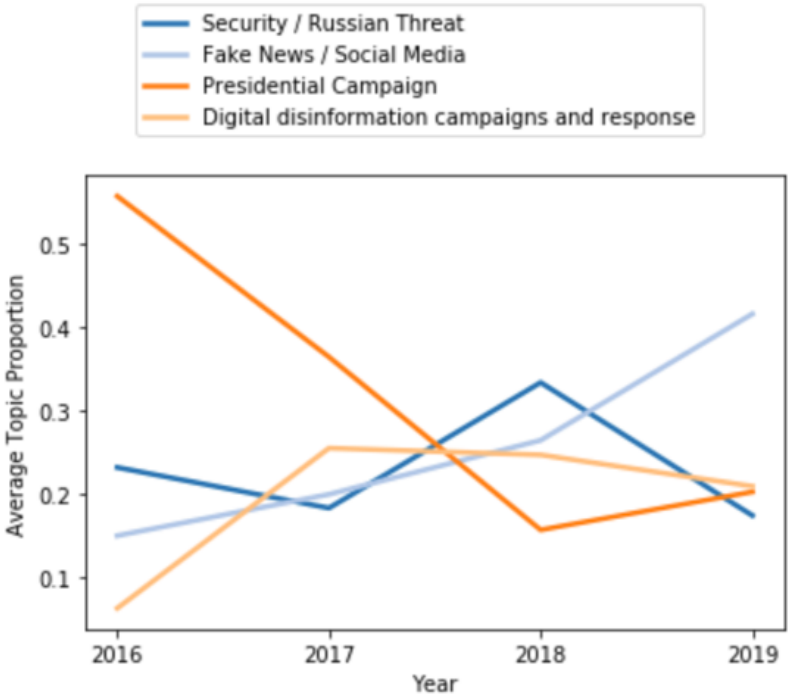


Figure 6: Evolution in Topic Prevalence - Validation Subset.

5 Conclusion

Using dynamic topic modeling, we were able to identify four dimensions along which elite actors might be conceptualizing issues surrounding digital propaganda. Each topic reveals a distinct aspect of computational propaganda and our time analysis shows how topic composition and topic prevalence changed over time. We observe a dynamic discursive space, with ongoing shifts in framing and in the content of the discourse.

The present study is limited in that the findings depend on subjectivity and that the results may not represent the entire think tank population. Our study is exploratory data analysis which requires subjective interpretation of the results. Consequently, we had to rely on our domain knowledge to make sense out of what the topic words mean. While such subjectivity is noteworthy, we were still able to draw meaningful and consistent interpretation from the results. Moreover, our corpus only includes reports from the thirteen think tanks out of 1871 in the U.S. Different results may come out if we fit the model with more reports from various think tanks. Nevertheless, examining the top twenty think tanks helps to get a glimpse of what might be a dominant view on computational propaganda although it is limited. For future studies, we may want to build a corpus by randomly selecting twenty think tanks and compare its results with the top twenty think tanks. In so doing, we might be able to show whether the results from the current study can be generalized to other think tank reports.

Appendices Begin After References

6 References

Atlantic Council), Matt Chessen (The. n.d. "The Madcom Future." https://www.atlanticcouncil.org/in-depth-research-reports/report/the-madcom-future.

Benoit, Kenneth, Kohei Watanabe, Haiyan Wang, Paul Nulty, Adam Obeng, Stefan Müller, and Akitaka Matsuo. 2018. "Quanteda: An R Package for the Quantitative Analysis of Textual Data." *Journal of Open Source Software* 3 (30): 774.

Blei, David M., and John D. Lafferty. 2016. "Dynamic Topic Models."

Blei, David M., Andrew Y. Ng, and Michael I. Jordan. 2003. "Latent Dirichlet Allocation." *Journal of Machine Learning Research*, no. 3: 993–1022.

DiMaggio, Paul. 2015. "Adapting Computational Text Analysis to Social Science (and Vice Versa)." *Big Data and Society* 2 (2).

Evans, James, and Pedro Aceves. 2016. "Machine Translation: Mining Text for Social Theory." *Review of Sociology* 42.

Grimmer, Justin, and Brandon M. Stewart. 2013. "Text as Data: The Promise and Pitfalls of Automatic Content Analysis Methods for Political Texts." *Political Analysis* 21 (3): 267–97.

Honnibal, Matthew, and Mark Johnson. 2015. "An Improved Non-Monotonic Transition System for Dependency Parsing." In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 1373–8. https://aclweb.org/anthology/D/D15/D15-1162: Association for Computational Linguistics.

Kollanyi, Bence, Philip Howard, and Samuel C Wolley. 2016. "Bots and Automation over Twitter During the Us Election." *Data Memo. Oxford, UK: Project on Computational Propaganda* 2016 (4): 1–5.

Lopwer, Edward, and Steven Bird. 2002. "NLTK; the Natural Language Toolkit." *NLTK; the Natural Language Toolkit.*

McGann, James G. 2019. *2018 Global Go to Think Tank Index Report.* University of Pennsylvania.

Mimno, David, Edmund Talley, Miriam Leenders, Hannah M. Wallach, and Andrew McCallum. 2011. "Optimizing Semantic Coherence in Topic Models." *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, 267–72.

Nelson, Laura K. 2017. "Computational Grounded Theory: A Methodoliogcal Framework." *Sociological Methods and Reseawrch*, 1–40.

Nelson, Laura K., Derek Burk, Marcel Knudsen, and Leslie McCall. 2018. "The Future of Coding: A Comparison of Hand-Coding and Three Types of Computer-Assisted Text Analysis Methods." *Sociological Methods and Research.*

Shinyama, Yusuke. 2019. "PDF Parser and Analyzer." http://github.com/euske/pdfminer.

Weiss, Brennan. 2018. "Russian Trolls Orchestrated Divisive Protests in the Us About Trump - Here Are 9 That We Know About." *Russian Trolls Orchestrated Divisive Protests in the US About Trump - Here Are 9 That We Know About* February 21.

Wolley, Samuel C, and Philip Howard. 2017. "Computational Propaganda Worldwide: Executive Summary." *The Computational Propaganda Project*.

Appendix A

Domain and Corpus Specific Stopwords

foreword	i
introduction	ii
acknowledgments	iii
conclusion	iv
endnotes	v
ibid.	vi
ibid	vii
background	viii
example	laughter
able	stinchfield
way	cei
summary	harvard
authors	kennedi
findings	center for american progress
recommendation	has
overview	was
applause	this
his	

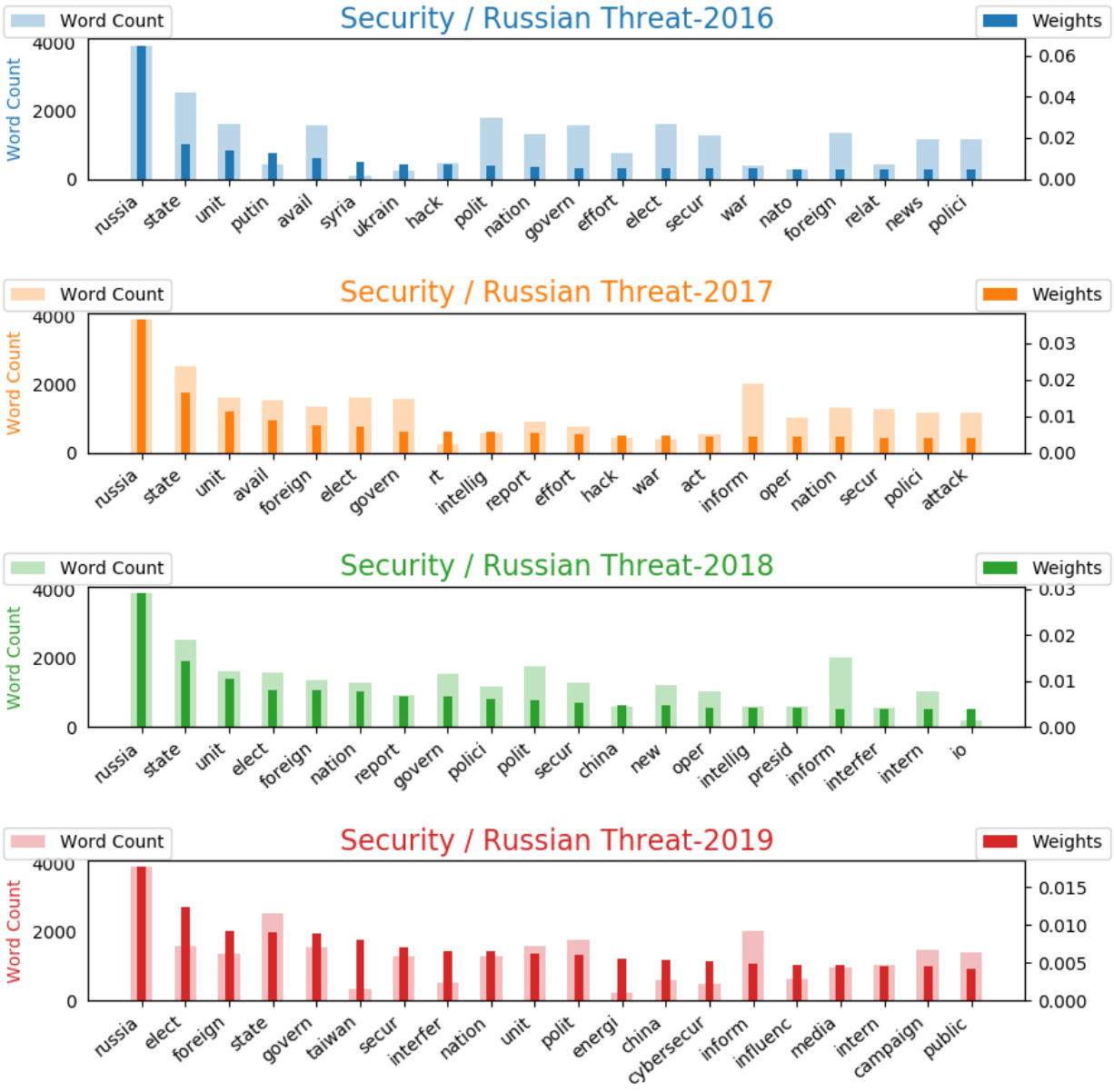
Appendix B

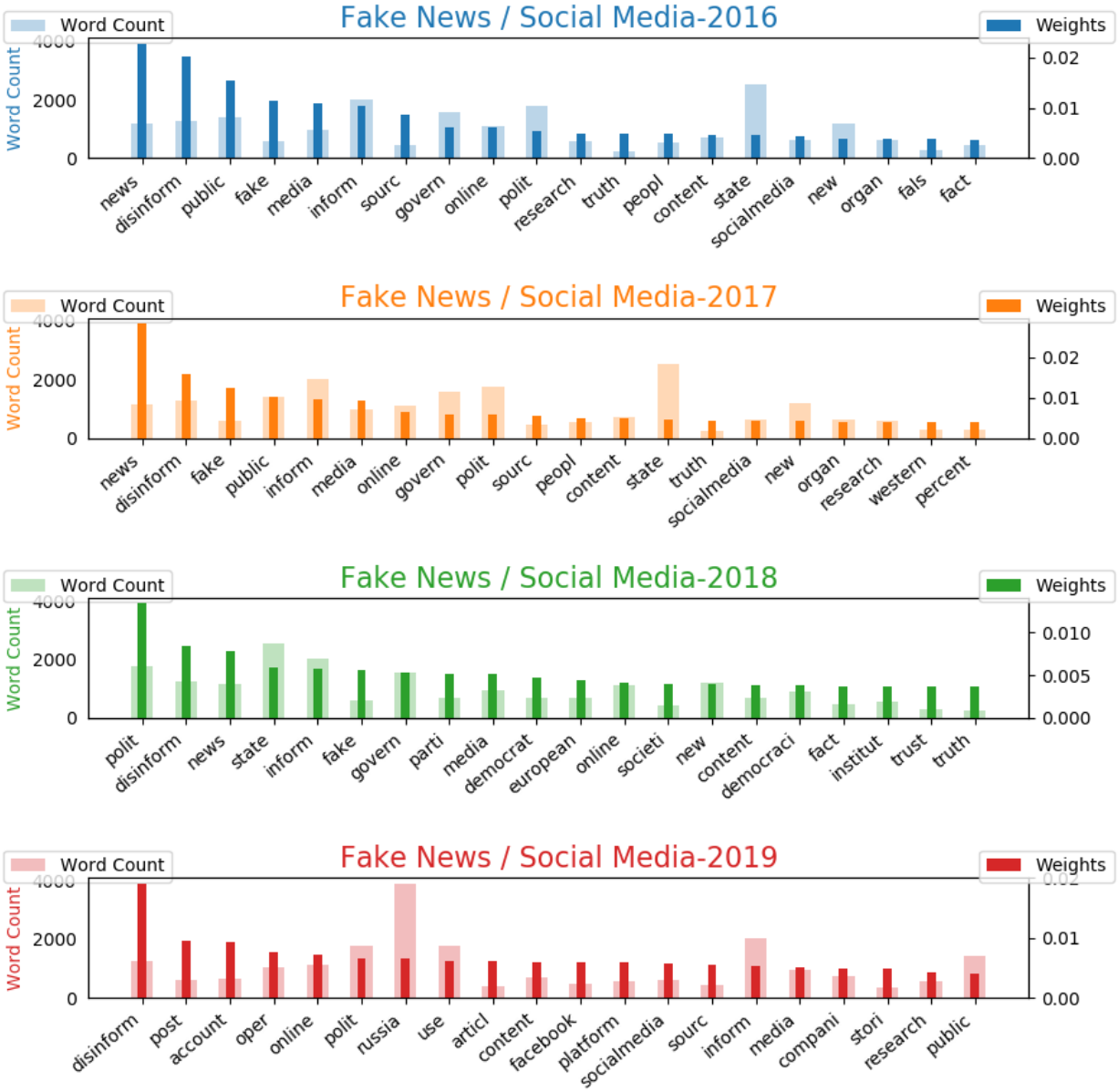
Joined and Transformed Words

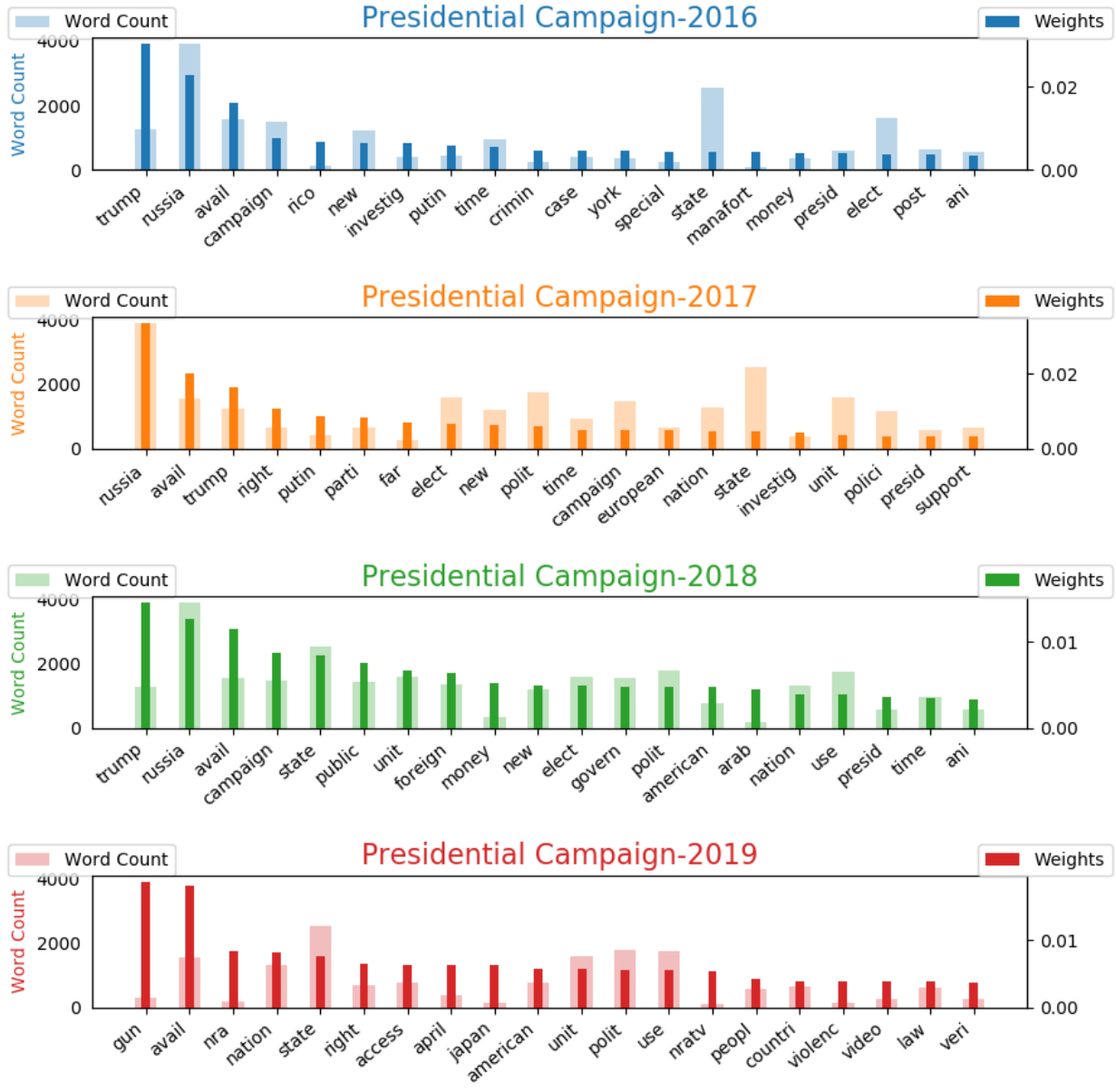
Input Word 1	Input Word 2	Output Word 2
donald	trump	**trump**
donald		**trump**
social	media	**social_media**
articial	intelligence	**ai**
machineeri		**machinelearning**
russian		**russia**
influence	operations	**influenceoperations**
cyber	security	**cybersecurity**

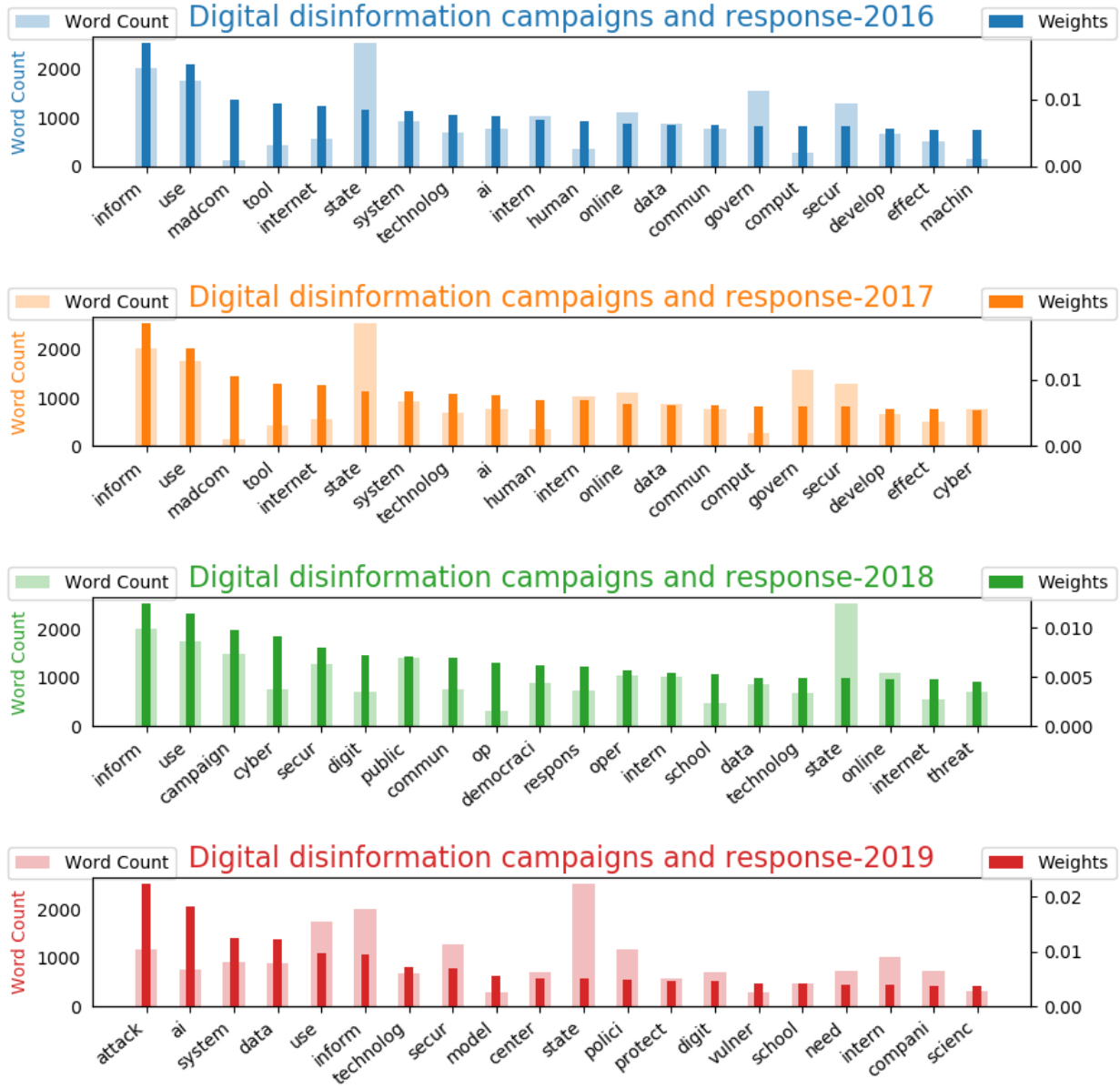
Appendix C

Topics and Keys



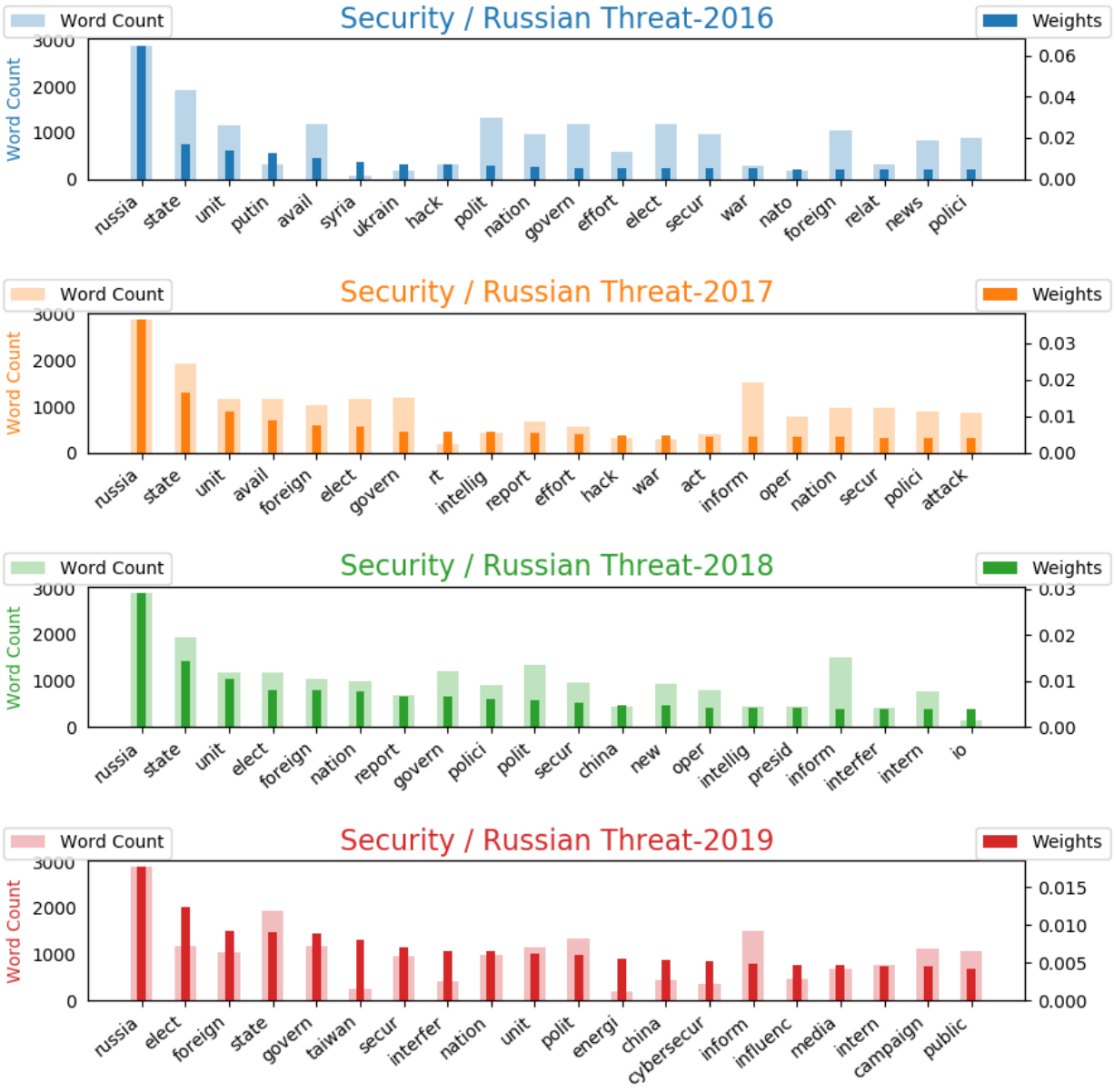

Fake News / Social Media-2016
Word Count
Weights
Word Count
4000
2000
0
0.02
0.01
0.00
news
disinform
public
fake
media
inform
sourc
govern
online
polit
research
truth
peopl
content
state
socialmedia
new
organ
fals
fact
Fake News / Social Media-2017
Word Count
Weights
Word Count
4000
2000
0
0.02
0.01
0.00
news
disinform
fake
public
inform
media
online
govern
polit
sourc
peopl
content
state
truth
socialmedia
new
organ
research
western
percent
Fake News / Social Media-2018
Word Count
Weights
Word Count
4000
2000
0
0.010
0.005
0.000
polit
disinform
news
state
inform
fake
govern
parti
media
democrat
european
online
societi
new
content
democraci
fact
institut
trust
truth
Fake News / Social Media-2019
Word Count
Weights
Word Count
4000
2000
0
0.01
0.00
disinform
post
account
oper
online
polit
russia
use
articl
content
facebook
platform
socialmedia
sourc
inform
media
compani
stori
research
public

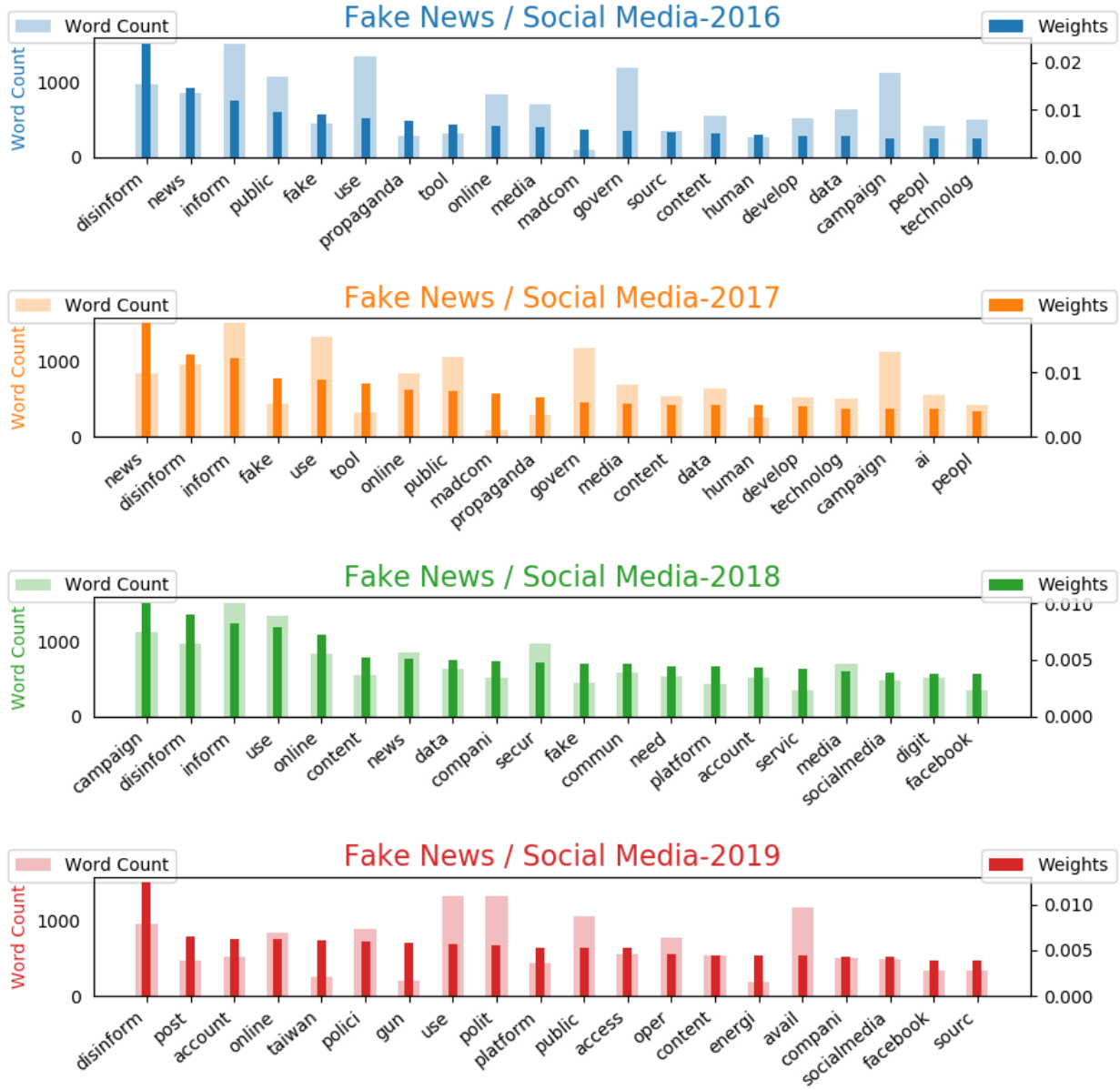

Presidential Campaign-2016
Word Count
Weights
Word Count
4000
2000
0
0.02
0.00
trump
russia
avail
campaign
rico
new
investig
putin
time
crimin
case
york
special
state
manafort
money
presid
elect
post
ani
Presidential Campaign-2017
Word Count
Weights
Word Count
4000
2000
0
0.02
0.00
russia
avail
trump
right
putin
parti
far
elect
new
polit
time
campaign
european
nation
state
investig
unit
polici
presid
support
Presidential Campaign-2018
Word Count
Weights
Word Count
4000
2000
0
0.01
0.00
trump
russia
avail
campaign
state
public
unit
foreign
money
new
elect
govern
polit
american
arab
nation
use
presid
time
ani
Presidential Campaign-2019
Word Count
Weights
Word Count
4000
2000
0
0.01
0.00
gun
avail
nra
nation
state
right
access
april
japan
american
unit
polit
use
nratv
peopl
countri
violenc
video
law
veri

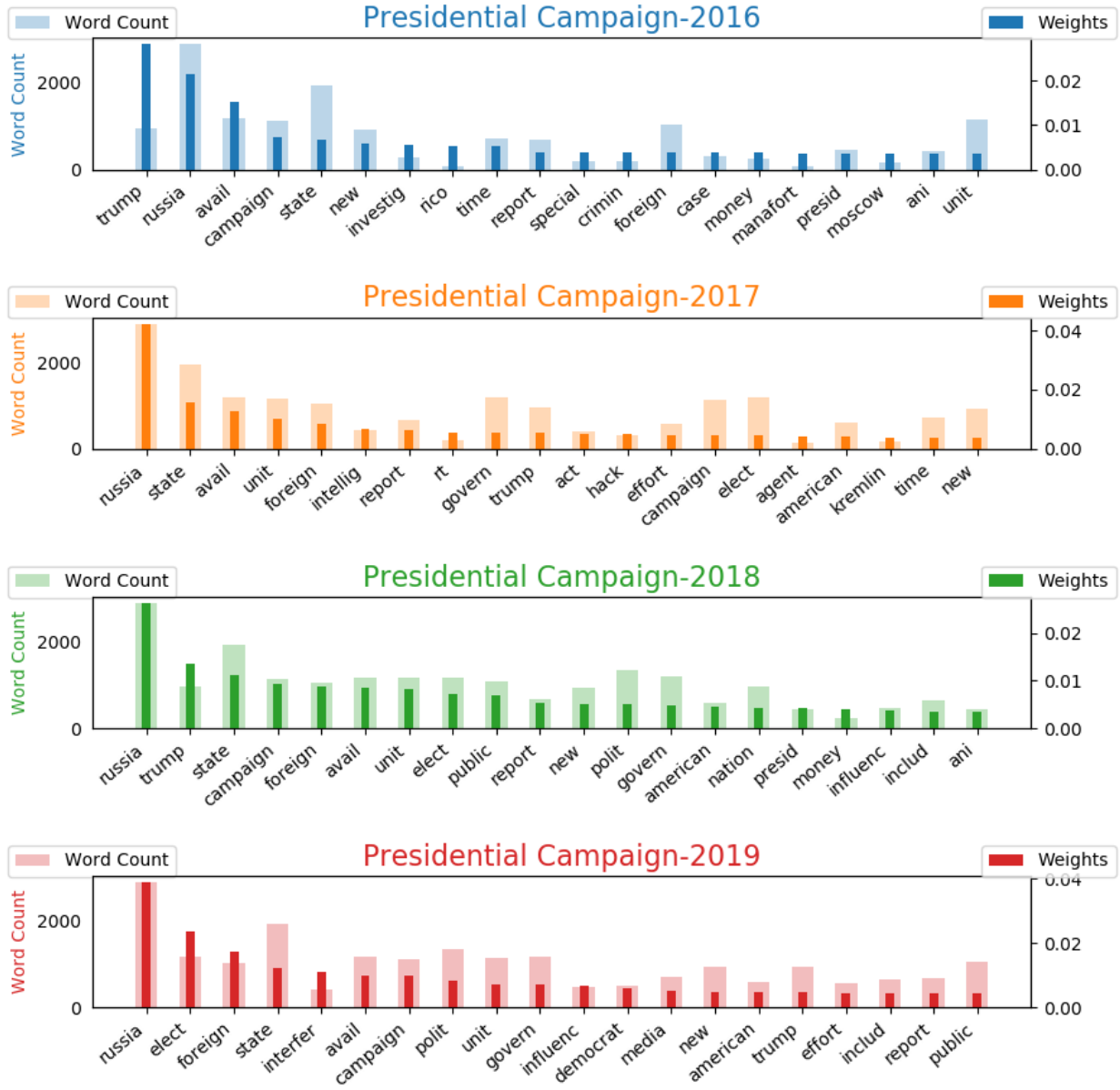

Word Count
Digital disinformation campaigns and response-2016
Weights
Word Count
2000
1000
0
0.01
0.00
inform
use
madcom
tool
internet
state
system
technolog
ai
intern
human
online
data
commun
govern
comput
secur
develop
effect
machin
Word Count
Digital disinformation campaigns and response-2017
Weights
Word Count
2000
1000
0
0.01
0.00
inform
use
madcom
tool
internet
state
system
technolog
ai
human
intern
online
data
commun
comput
govern
secur
develop
effect
cyber
Word Count
Digital disinformation campaigns and response-2018
Weights
Word Count
2000
1000
0
0.010
0.005
0.000
inform
use
campaign
cyber
secur
digit
public
commun
op
democraci
respons
oper
intern
school
data
technolog
state
online
internet
threat
Word Count
Digital disinformation campaigns and response-2019
Weights
Word Count
2000
1000
0
0.02
0.01
0.00
attack
ai
system
data
use
inform
technolog
secur
model
center
state
polici
protect
digit
vulner
school
need
intern
compani
scienc

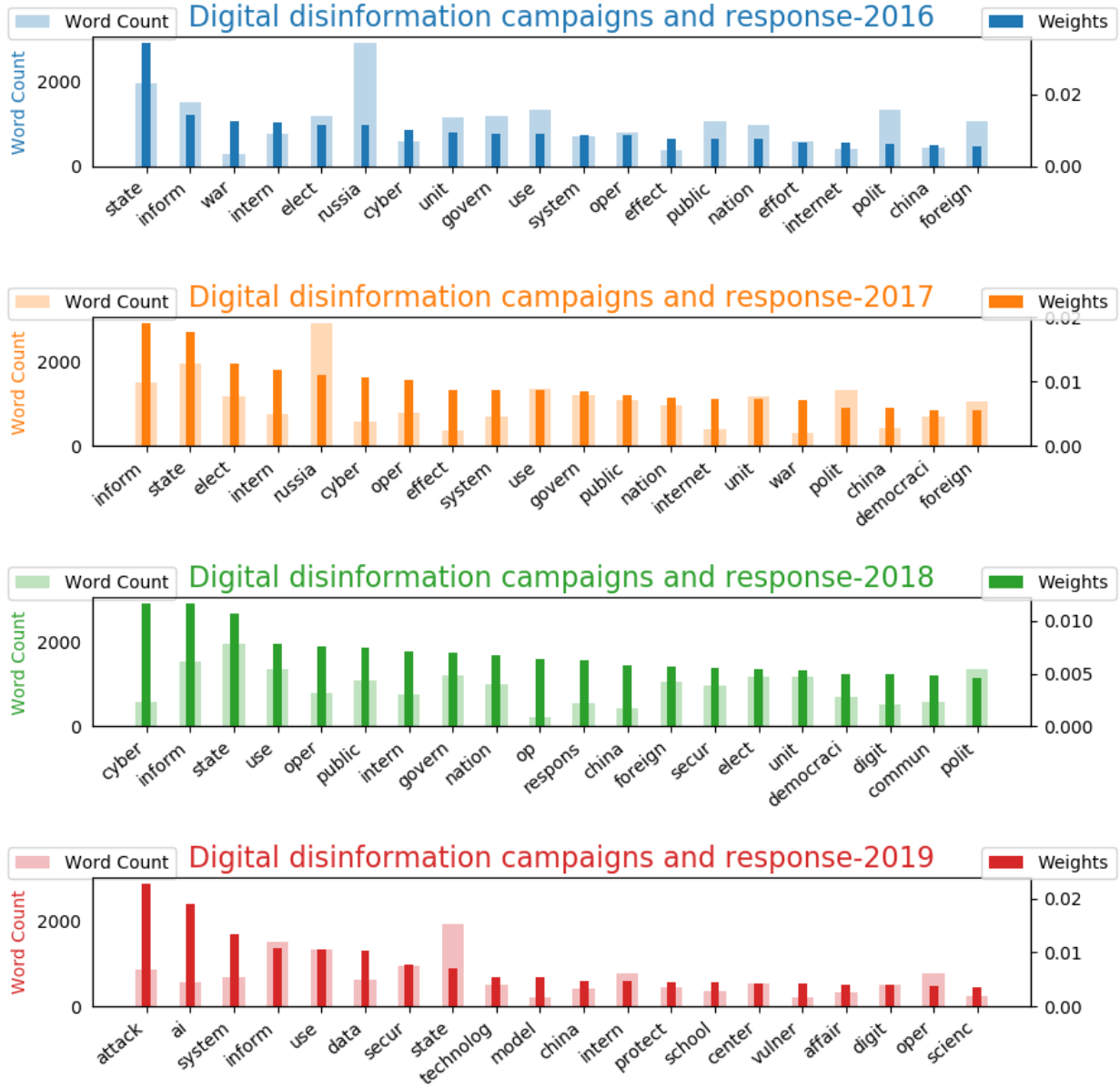
Appendix D

Topics and Keys - Validation Set




Fake News / Social Media-2016
Word Count
Weights
Word Count
1000
0
0.02
0.01
0.00
disinform
news
inform
public
fake
use
propaganda
tool
online
media
madcom
govern
sourc
content
human
develop
data
campaign
peopl
technolog
Fake News / Social Media-2017
Word Count
Weights
Word Count
1000
0
0.01
0.00
news
disinform
inform
fake
use
tool
online
public
madcom
propaganda
govern
media
content
data
human
develop
technolog
campaign
ai
peopl
Fake News / Social Media-2018
Word Count
Weights
Word Count
1000
0
0.005
0.000
campaign
disinform
inform
use
online
content
news
data
compani
secur
fake
commun
need
platform
account
servic
media
socialmedia
digit
facebook
Fake News / Social Media-2019
Word Count
Weights
Word Count
1000
0
0.010
0.005
0.000
disinform
post
account
online
taiwan
polici
gun
use
polit
platform
public
access
oper
content
energi
avail
compani
socialmedia
facebook
sourc


Presidential Campaign-2016
Word Count
Weights
Word Count
2000
0
0.02
0.01
0.00
trump
russia
avail
campaign
state
new
investig
rico
time
report
special
crimin
foreign
case
money
manafort
presid
moscow
ani
unit
Presidential Campaign-2017
Word Count
Weights
Word Count
2000
0
0.04
0.02
0.00
russia
state
avail
unit
foreign
intellig
report
rt
govern
trump
act
hack
effort
campaign
elect
agent
american
kremlin
time
new
Presidential Campaign-2018
Word Count
Weights
Word Count
2000
0
0.02
0.01
0.00
russia
trump
state
campaign
foreign
avail
unit
elect
public
report
new
polit
govern
american
nation
presid
money
influenc
includ
ani
Presidential Campaign-2019
Word Count
Weights
Word Count
2000
0
0.04
0.02
0.00
russia
elect
foreign
state
interfer
avail
campaign
polit
unit
govern
influenc
democrat
media
new
american
trump
effort
includ
report
public


Word Count
Digital disinformation campaigns and response-2016
Weights
Word Count
2000
0
0.02
0.00
state
inform
war
intern
elect
russia
cyber
unit
govern
use
system
oper
effect
public
nation
effort
internet
polit
china
foreign
Word Count
Digital disinformation campaigns and response-2017
Weights
Word Count
2000
0
0.02
0.01
0.00
inform
state
elect
intern
russia
cyber
oper
effect
system
use
govern
public
nation
internet
unit
war
polit
china
democraci
foreign
Word Count
Digital disinformation campaigns and response-2018
Weights
Word Count
2000
0
0.010
0.005
0.000
cyber
inform
state
use
oper
public
intern
govern
nation
op
respons
china
foreign
secur
elect
unit
democraci
digit
commun
polit
Word Count
Digital disinformation campaigns and response-2019
Weights
Word Count
2000
0
0.02
0.01
0.00
attack
ai
system
inform
use
data
secur
state
technolog
model
china
intern
protect
school
center
vulner
affair
digit
oper
scienc

Code Begins on Next Page

dtm_analysis_final

December 10, 2019

0.1 Dynamic Topic Modeling Analysis

[1]:
```
# import packages
import pandas as pd
import os
import csv
import numpy as np
import gensim
import gensim.corpora as corpora
from gensim.models import ldaseqmodel
import time
import pickle
from gensim.models.coherencemodel import CoherenceModel
import matplotlib.pyplot as plt
from collections import Counter
import matplotlib.colors as mcolors
```

0.1.1 1. Read in texts and build a data frame version

[2]:
```
directory = "/Users/ditong/Documents/uml/project/final_corpus/"
folders = os.listdir(directory)
```

[3]:
```
df = pd.DataFrame({'filename': [], 'publisher': [], 'year': [], 'text': []})
for folder in folders:
    if not folder.startswith('.'):
        files = os.listdir(directory+ '/' + folder)
        for file in files:
            if not file.startswith('.'):
                with open(directory+ '/' + folder + '/' + file,␣
↪encoding='ISO-8859-1') as f:
                    df = df.append({'filename': file, 'publisher': folder,␣
↪'year': file[:4],
                                    'text': f.read()}, ignore_index=True)
```

[5]:
```
pd.DataFrame(df).to_csv("/Users/ditong/Documents/uml/project/data_final.csv")
```

0.1.2 2. Tokenize the texts, build the corpus and set the time silce

[4]:
```
# read in csv file
data = pd.read_csv('./data_final.csv')
# sort by year
sorted_data = data.sort_values(by=['year'])
# change index
sorted_data = sorted_data.reset_index()
sorted_data = sorted_data.drop(columns=['index', 'Unnamed: 0'])
```

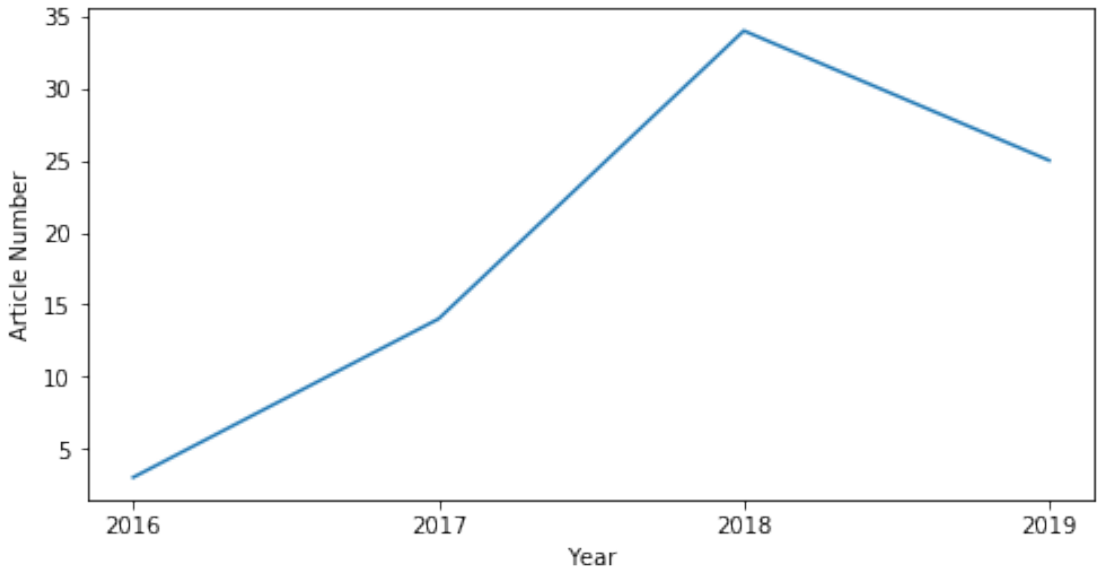
[5]:
```
# construct the function that preprocess the text
def preprocess(text):
    '''
    Preprocess the text by tokenizing the string into uni-grams, deleting all
    numbers, punctuations and stop words. Store the preprocessed sting into
    a list of words

    input:
        text: text for preprocessing(str)
    output: a list of words
    '''
    result = []
    tokens = gensim.utils.tokenize(text)
    tokens_lst = list(tokens)

    return tokens_lst
```

[6]:
```
# preprocess the data
processed_df = sorted_data['text'].map(preprocess)
```

[7]:
```
# prepare Document-Term Matrix for the DTA model
# Create Dictionaries for unique word counts of each decade
dic_all = corpora.Dictionary(processed_df)

# Create Corpus: Term Document Frequency
corpus_all = [dic_all.doc2bow(text) for text in processed_df]
```

[13]:
```
# set the time slice
sorted_data[sorted_data['year']==2016]
# 2016: 0-2
sorted_data[sorted_data['year']==2017]
# 2017: 3-16
sorted_data[sorted_data['year']==2018]
# 2018: 17-50
sorted_data[sorted_data['year']==2019]
# 2019: 51-75
time_slice = [3, 14, 34, 25]
```

[15]:
```
plt.figure(figsize=(8, 4))
plt.plot([int(2016), int(2017), int(2018), int(2019)], time_slice)
plt.xticks(np.arange(2016, 2020))
plt.xlabel("Year")
plt.ylabel("Article Number")
plt.savefig('./visualization_3/article_number_evolution.png')
plt.show()
```



0.1.3 3. Fit six models with topic numbers 4-10, report results and topic conherence score

[12]:
```
# fit the 4-topic model
start = time.time()
ldaseq_4 = ldaseqmodel.LdaSeqModel(corpus=corpus_all, id2word=dic_all,␣
 ↪time_slice=time_slice,
                                    num_topics=4, chain_variance=0.1)
end = time.time()
print(end - start)
```

```
/Users/ditong/anaconda3/lib/python3.7/site-
packages/gensim/models/ldaseqmodel.py:230: RuntimeWarning: divide by zero
encountered in double_scalars
  convergence = np.fabs((bound - old_bound) / old_bound)

1963.5981812477112
```

[13]:
```
# save model
pickle.dump(ldaseq_4, open("ldaseq_model_4_final.sav", 'wb'))
```

[14]:
```
# calculate coherence matrix
topics_dtm_41 = ldaseq_4.dtm_coherence(time=0)
topics_dtm_42 = ldaseq_4.dtm_coherence(time=1)
topics_dtm_43 = ldaseq_4.dtm_coherence(time=2)
topics_dtm_44 = ldaseq_4.dtm_coherence(time=3)
cm_DTM_41 = CoherenceModel(topics=topics_dtm_41, corpus=corpus_all,␣
 ↪dictionary=dic_all, coherence='u_mass')
cm_DTM_42 = CoherenceModel(topics=topics_dtm_42, corpus=corpus_all,␣
 ↪dictionary=dic_all, coherence='u_mass')
cm_DTM_43 = CoherenceModel(topics=topics_dtm_43, corpus=corpus_all,␣
 ↪dictionary=dic_all, coherence='u_mass')
cm_DTM_44 = CoherenceModel(topics=topics_dtm_44, corpus=corpus_all,␣
 ↪dictionary=dic_all, coherence='u_mass')
print ("U_mass topic coherence")
print ("DTM Python coherence is", cm_DTM_41.get_coherence(), ";", cm_DTM_42.
 ↪get_coherence(), ";",
      cm_DTM_43.get_coherence(), ";", cm_DTM_44.get_coherence())
```

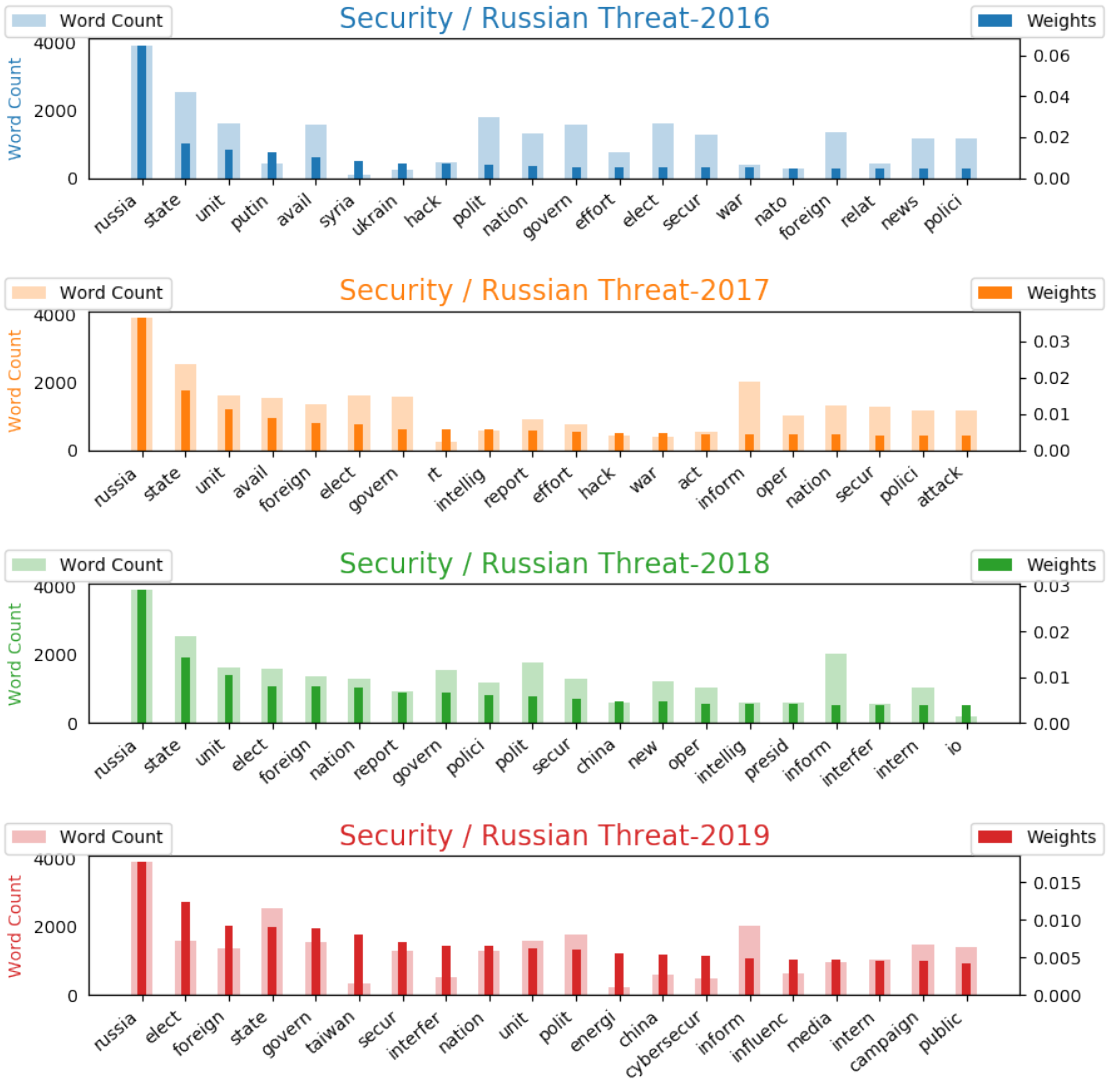
```
U_mass topic coherence
DTM Python coherence is -0.2687402905479692 ; -0.20485455161750746 ;
-0.2707217104978128 ; -0.3760027881470409
```

[11]:
```
# print topic evolution
first_topic_4 = ldaseq_4.print_topic_times(topic=0)
second_topic_4 = ldaseq_4.print_topic_times(topic=1)
third_topic_4 = ldaseq_4.print_topic_times(topic=2)
fourth_topic_4 = ldaseq_4.print_topic_times(topic=3)
```

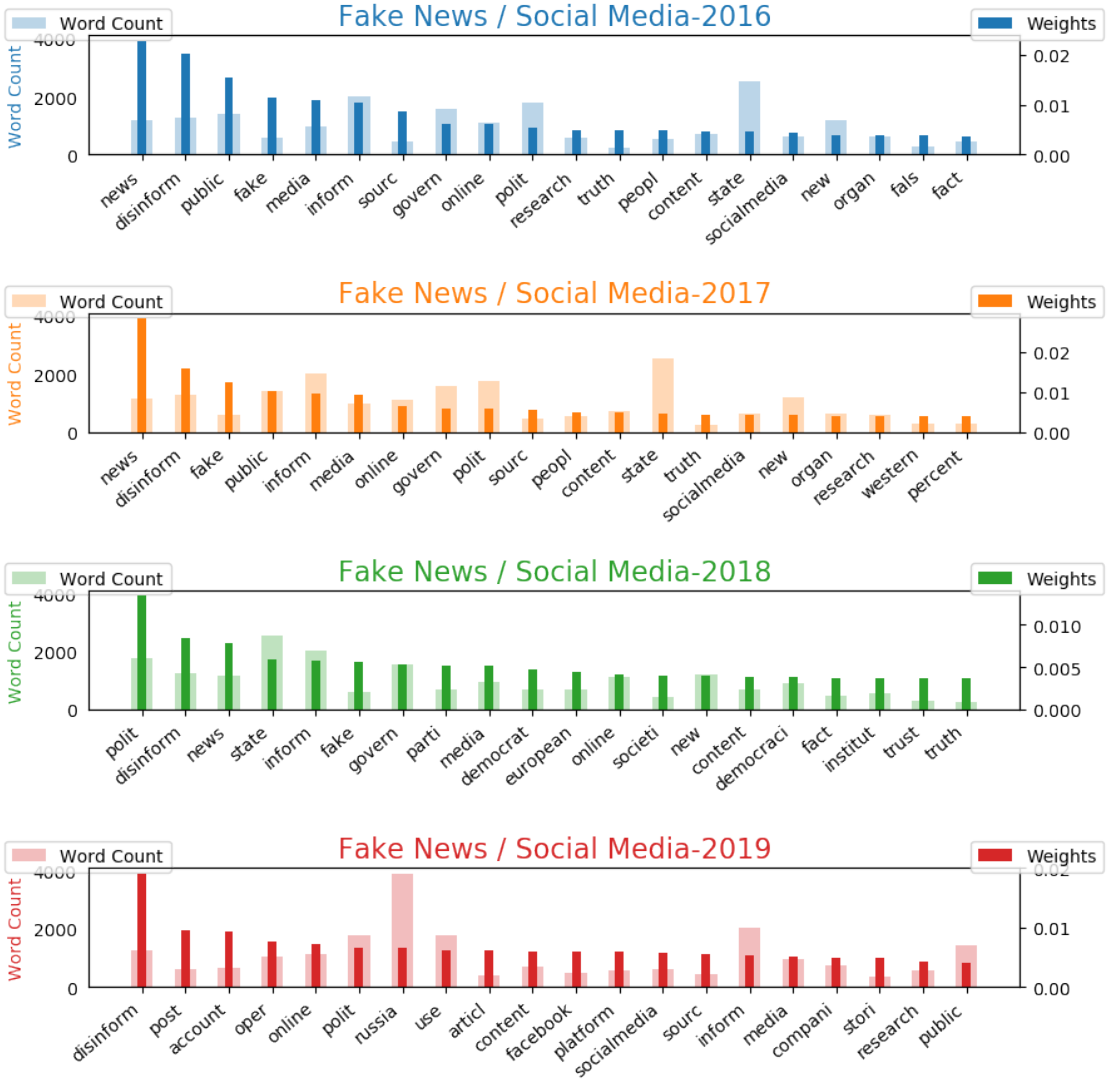
[15]:
```
# build the function that construct the dataframe for visualization
def construct_vis_df(dataframe, topics):
    # count how frequently a word appears in the document
    data_flat = [w for w_list in dataframe for w in w_list]
    counter = Counter(data_flat)
    # store the word, topic, word weight and word count into a list
    out = []
    for i in range(len(topics)):
        for word, weight in topics[i]:
            out.append([word, i, weight, counter[word]])
    # transform the list into a pandas dataframe
    df = pd.DataFrame(out, columns=['word', 'time_id', 'importance',␣
 ↪'word_count'])

    return df
```

[16]:
```
# construct the function to graph Word Counts of Topic Keywords for all
# topics (reference: https://www.machinelearningplus.com/nlp/
 ↪topic-modeling-visualization-how-to-present-results-lda-models/)
def graph_topic_keyword_count(df, topic_name, time_period):
    '''
    Build graphs for the ten topics identified by the lda model based on the
    dataframe, each graph present the counts and weights of the keywords of
    the topic, save the graphs in the visualization folder.

    input:
        model: the LDA model that identifies the ten topics based on the␣
 ↪dataframe
        dataframe: the dataframe used by the LDA model to identify topics
        time_period: (str) the year of the dataframe (e.g. '2016')
    '''

    # Plot Word Count and Weights of Topic Keywords
    fig, axes = plt.subplots(4, 1, figsize=(9,9), sharey=True, dpi=100)
    cols = [color for name, color in mcolors.TABLEAU_COLORS.items()]
    for i, ax in enumerate(axes.flatten()):
        ax.bar(x='word', height="word_count", data=df.loc[df.time_id==i, :],\
                color=cols[i], width=0.5, alpha=0.3, label='Word Count')
        ax_twin = ax.twinx()
        ax_twin.bar(x='word', height="importance", data=df.loc[df.time_id==i, :
 ↪],\
                    color=cols[i], width=0.2, label='Weights')
        ax.set_ylabel('Word Count', color=cols[i])
        ax.set_title(time_period[i], color=cols[i], fontsize=16)
        ax.tick_params(axis='y', left=False)
        ax.set_xticklabels(df.loc[df.time_id==i, 'word'], rotation=40, \
                           horizontalalignment= 'right')
        ax.legend(loc='upper left', bbox_to_anchor=(-0.1, 1.3))
        ax_twin.legend(loc='upper right', bbox_to_anchor=(1.1, 1.3))

    fig.tight_layout(w_pad=2)
    # save the graphs as files in the visualization folder
    plt.savefig('./visualization_final/{}_topic_keyword_count_importance'.
 ↪format(topic_name))
```

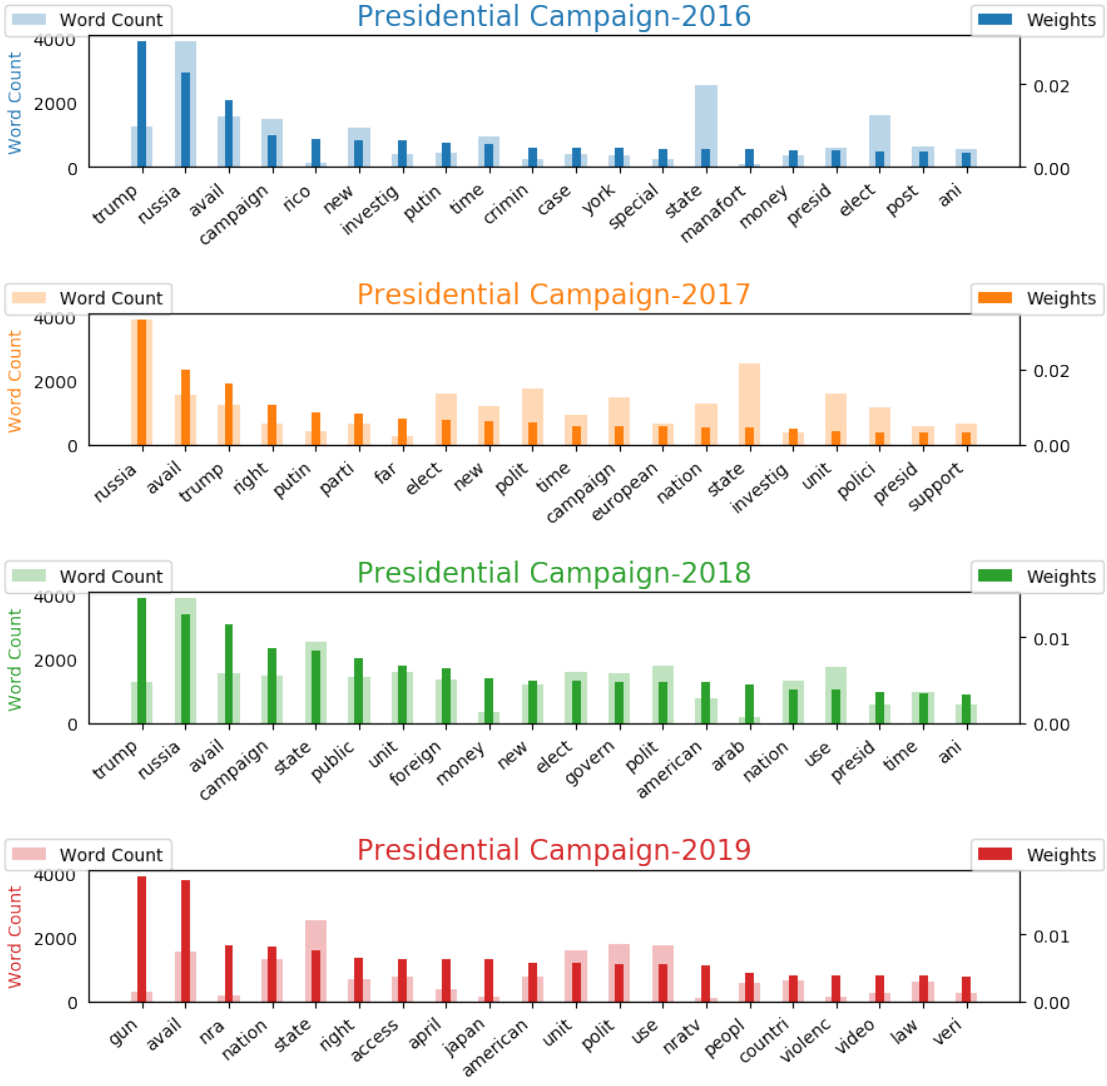
[17]:
```
#the first topic
vis_df_41 = construct_vis_df(processed_df, first_topic_4)
times_41 = ["Security / Russian Threat-2016", "Security / Russian Threat-2017",
         "Security / Russian Threat-2018", "Security / Russian Threat-2019"]
graph_topic_keyword_count(vis_df_41, 'Topic1_Security_Russian Threat', times_41)
```



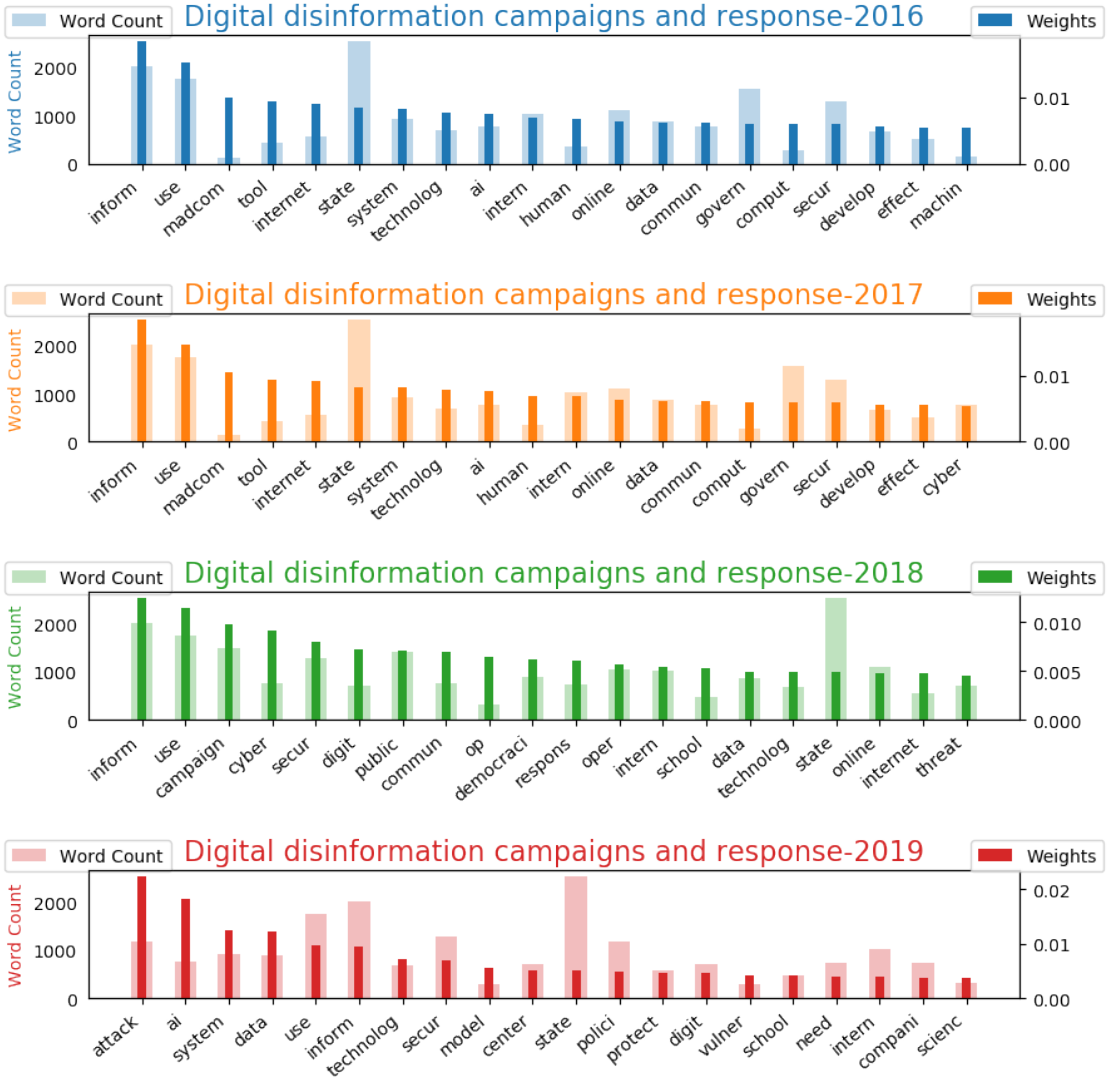
[18]:
```
#the second topic
vis_df_42 = construct_vis_df(processed_df, second_topic_4)
times_42 = ["Fake News / Social Media-2016", "Fake News / Social Media-2017",
        "Fake News / Social Media-2018", "Fake News / Social Media-2019"]
graph_topic_keyword_count(vis_df_42, 'Topic2_FakeNews_SocialMedia', times_42)
```



[19]:
```
#the third topic
vis_df_43 = construct_vis_df(processed_df, third_topic_4)
times_43 = ["Presidential Campaign-2016", "Presidential Campaign-2017",
        "Presidential Campaign-2018", "Presidential Campaign-2019"]
graph_topic_keyword_count(vis_df_43, 'Topic3_Presidential_Campaign', times_43)
```



[20]:
```
#the fourth topic
vis_df_44 = construct_vis_df(processed_df, fourth_topic_4)
times_44 = ["Digital disinformation campaigns and response-2016", "Digital␣
 ↪disinformation campaigns and response-2017",
        "Digital disinformation campaigns and response-2018", "Digital␣
 ↪disinformation campaigns and response-2019"]
graph_topic_keyword_count(vis_df_44,␣
 ↪'Topic4_Digital_disinformation_campaigns_and_response', times_44)
```

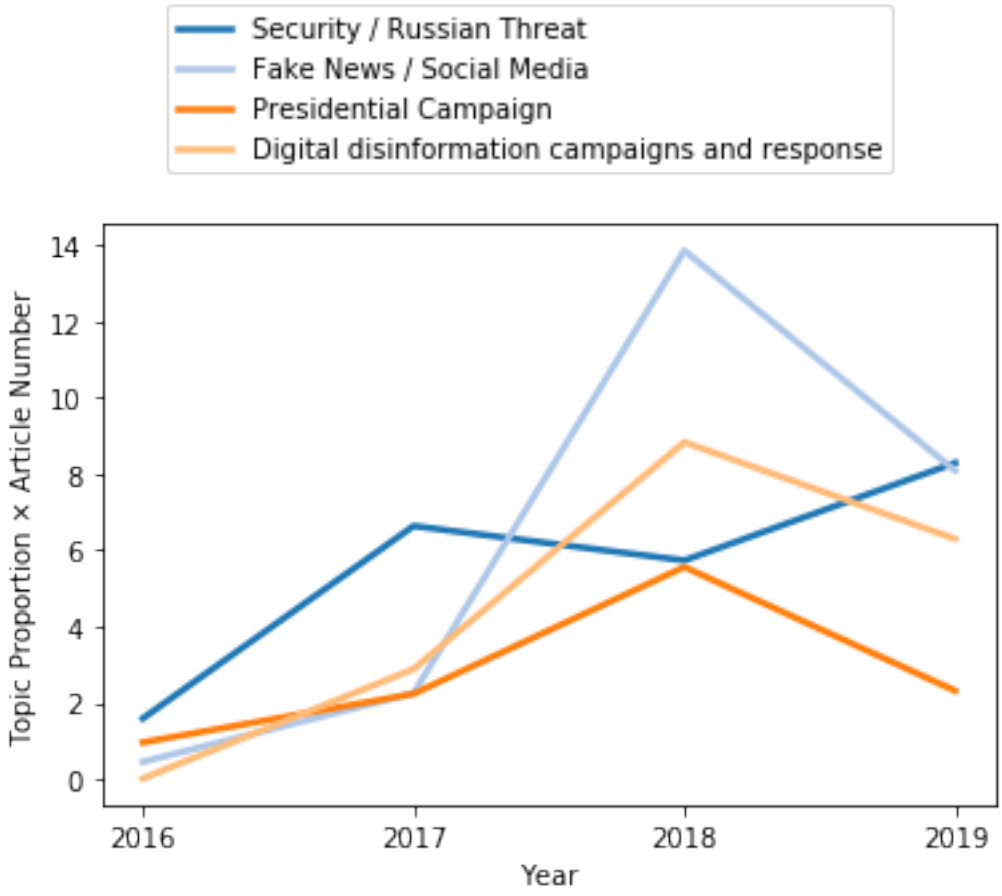


[12]:
```
doc_topic_4 = []
for doc_num in range(len(sorted_data)):
    doc = ldaseq_4.doc_topics(doc_num)
    doc_topic_4.append(doc)
topic_df_4 = pd.DataFrame(doc_topic_4, columns=['t1', 't2', 't3', 't4'])
topic_com_df_4 = pd.concat([sorted_data, topic_df_4], axis=1, sort=False)
grouped_df_4 = topic_com_df_4.groupby(['year']).sum().reset_index()
grouped_df_av_4 = topic_com_df_4.groupby(['year']).mean().reset_index()
```

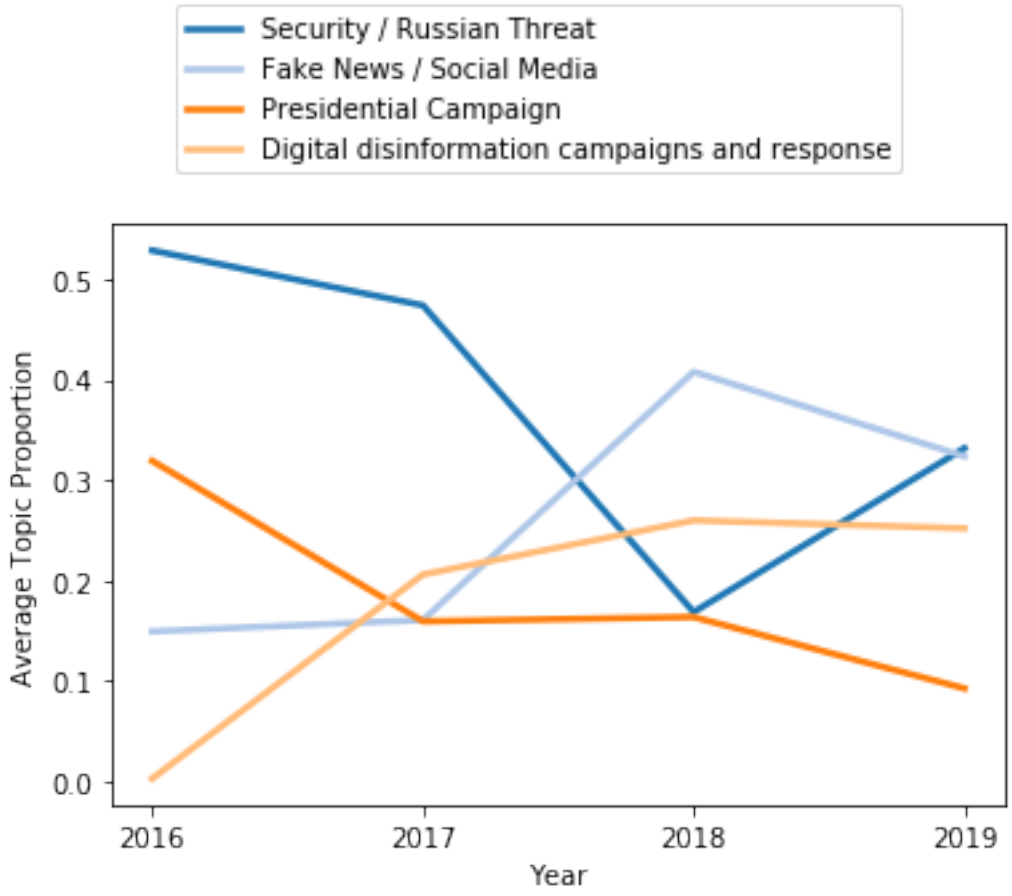
[7]:
```
def topic_prevalence_evolution(df, topics, labels, ylabel):

    color_sequence = ['#1f77b4', '#aec7e8', '#ff7f0e', '#ffbb78', '#2ca02c',
                      '#98df8a', '#d62728', '#ff9896', '#9467bd', '#c5b0d5',
                      '#8c564b', '#c49c94', '#e377c2', '#f7b6d2', '#7f7f7f',
```

```
                  '#c7c7c7', '#bcbd22', '#dbdb8d', '#17becf', '#9edae5']
    plt.figure(figsize=(6, 4))
    for i in range(len(labels)):
        plt.plot(grouped_df_4['year'], df[topics[i]], label=labels[i], color =␣
 ↪color_sequence[i], linewidth=2.5)
    plt.xticks(np.arange(2016, 2020))
    plt.xlabel("Year")
    plt.ylabel(ylabel)
    plt.legend(bbox_to_anchor=(0.9, 1.4))
    plt.savefig('./visualization_final/topic_prevalence_evolution.png')
    plt.show()
```

[49]:
```
topics = ['t1', 't2', 't3', 't4']
labels = ['Security / Russian Threat', 'Fake News / Social Media',␣
 ↪'Presidential Campaign',
          'Digital disinformation campaigns and response']
ylabel = "Topic Proportion ℰ Article Number"
topic_prevalence_evolution(grouped_df_4, topics, labels, ylabel)
```



[48]:
```
ylabel = "Average Topic Proportion"
topic_prevalence_evolution(grouped_df_av_4, topics, labels, ylabel)
```



[50]:
```
# fit the 5-topic model
start = time.time()
ldaseq_5 = ldaseqmodel.LdaSeqModel(corpus=corpus_all, id2word=dic_all,␣
 ↪time_slice=time_slice,
                                    num_topics=5, chain_variance=0.1)
end = time.time()
print(end - start)
```

```
/Users/ditong/anaconda3/lib/python3.7/site-
packages/gensim/models/ldaseqmodel.py:230: RuntimeWarning: divide by zero
encountered in double_scalars
  convergence = np.fabs((bound - old_bound) / old_bound)

2355.7922999858856
```

[51]:
```
# save model
pickle.dump(ldaseq_5, open("ldaseq_model_5_new.sav", 'wb'))
```

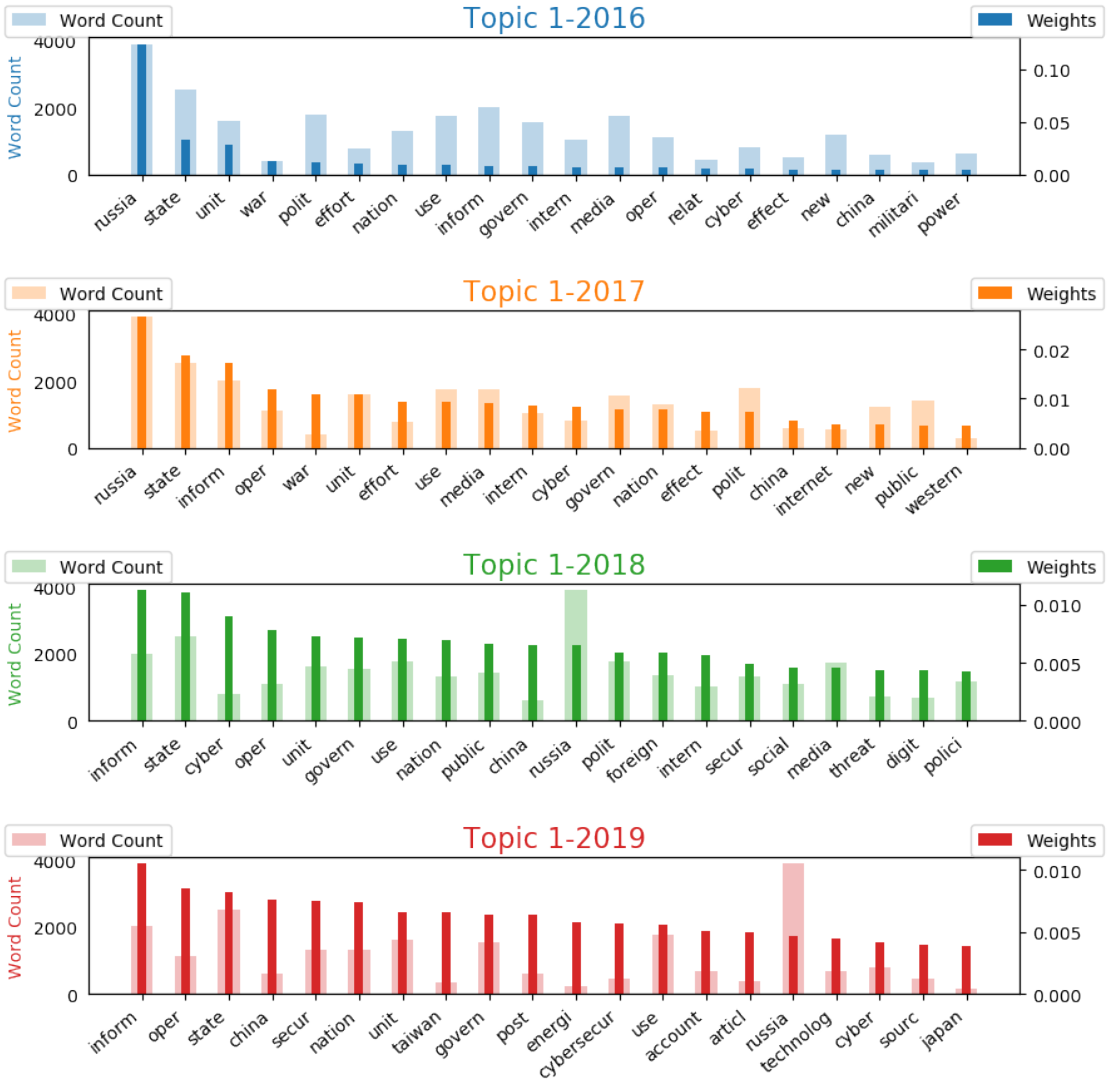
[10]:
```
# load model
ldaseq_5 = pickle.load(open("ldaseq_model_5.sav", 'rb'))
```

[52]:
```
# calculate coherence matrix
topics_dtm_51 = ldaseq_5.dtm_coherence(time=0)
topics_dtm_52 = ldaseq_5.dtm_coherence(time=1)
topics_dtm_53 = ldaseq_5.dtm_coherence(time=2)
topics_dtm_54 = ldaseq_5.dtm_coherence(time=3)
cm_DTM_51 = CoherenceModel(topics=topics_dtm_51, corpus=corpus_all,␣
 ↪dictionary=dic_all, coherence='u_mass')
cm_DTM_52 = CoherenceModel(topics=topics_dtm_52, corpus=corpus_all,␣
 ↪dictionary=dic_all, coherence='u_mass')
cm_DTM_53 = CoherenceModel(topics=topics_dtm_53, corpus=corpus_all,␣
 ↪dictionary=dic_all, coherence='u_mass')
cm_DTM_54 = CoherenceModel(topics=topics_dtm_54, corpus=corpus_all,␣
 ↪dictionary=dic_all, coherence='u_mass')
print ("U_mass topic coherence")
print ("DTM Python coherence is", cm_DTM_51.get_coherence(), ";", cm_DTM_52.
 ↪get_coherence(), ";",
      cm_DTM_53.get_coherence(), ";", cm_DTM_54.get_coherence())
```

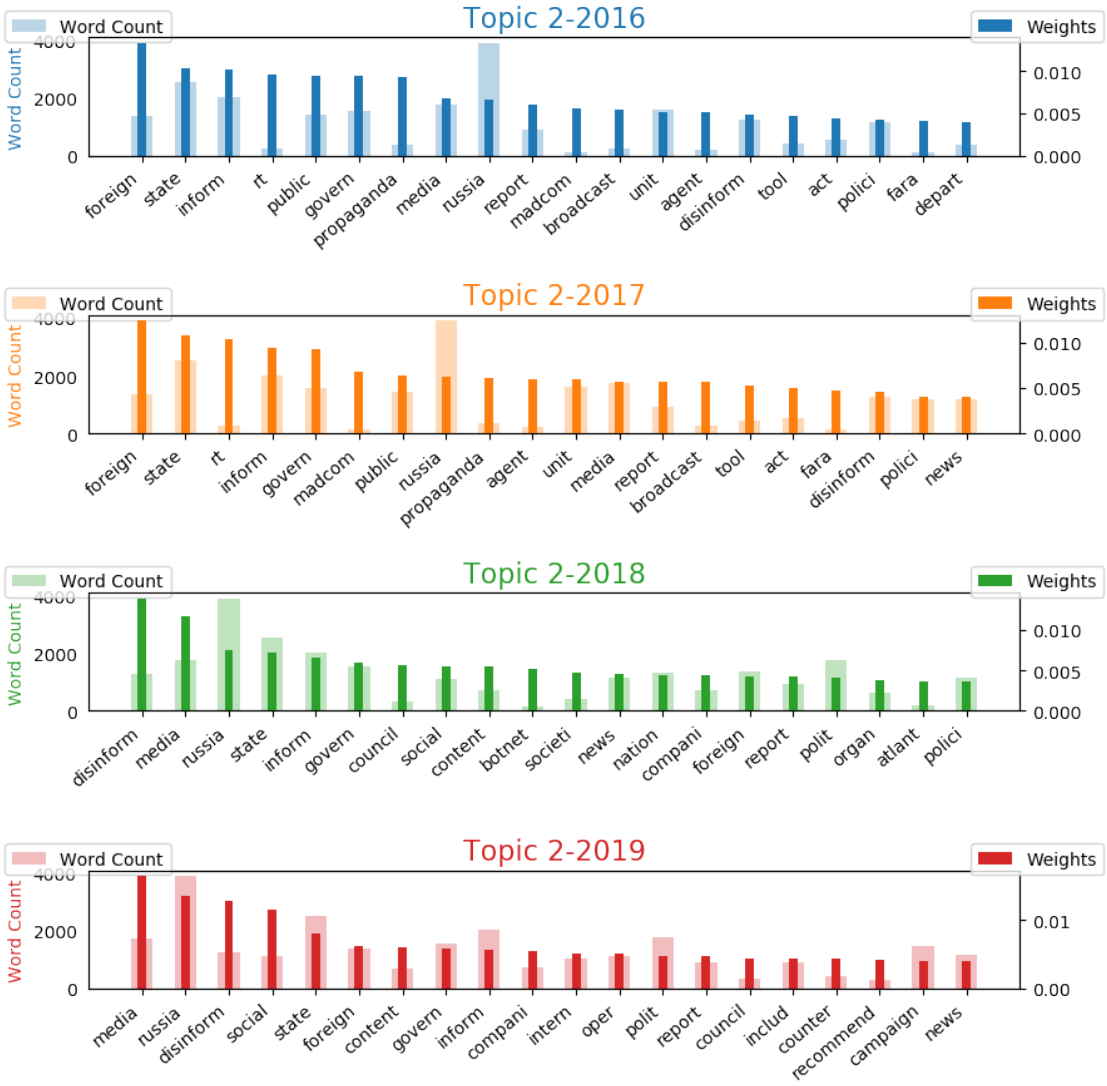
```
U_mass topic coherence
DTM Python coherence is -0.3447232545553641 ; -0.2796197189228443 ;
-0.26063024689608677 ; -0.347210590444595
```

[53]:
```
# print topic evolution
first_topic_5 = ldaseq_5.print_topic_times(topic=0)
second_topic_5 = ldaseq_5.print_topic_times(topic=1)
third_topic_5 = ldaseq_5.print_topic_times(topic=2)
fourth_topic_5 = ldaseq_5.print_topic_times(topic=3)
fifth_topic_5 = ldaseq_5.print_topic_times(topic=4)
```

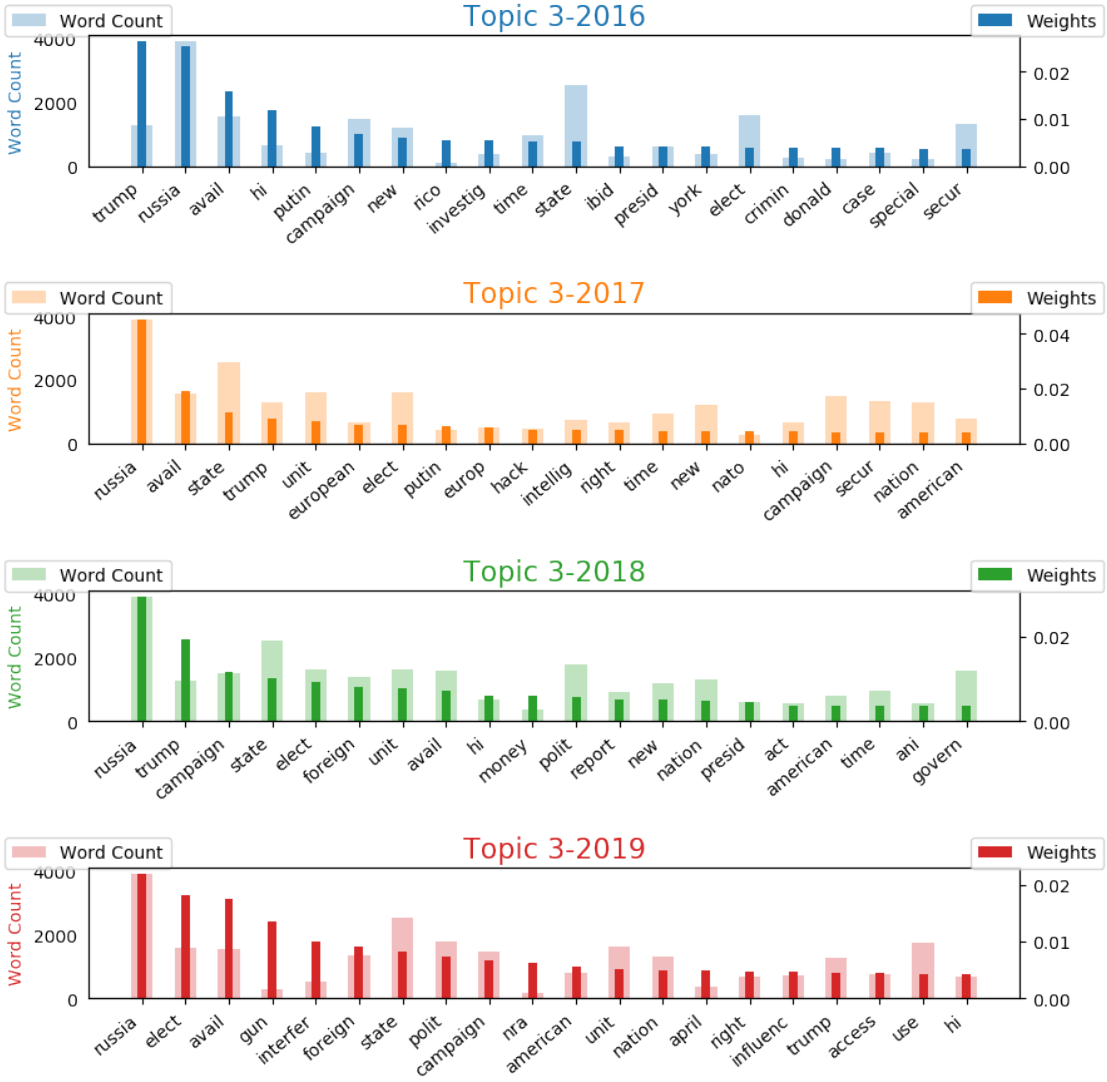
[57]:
```
#the first topic
vis_df_1 = construct_vis_df(processed_df, first_topic_5)
times_1 = ["Topic 1-2016", "Topic 1-2017",
         "Topic 1-2018", "Topic 1-2019"]
graph_topic_keyword_count(vis_df_1, 'Topic1', times_1)
```



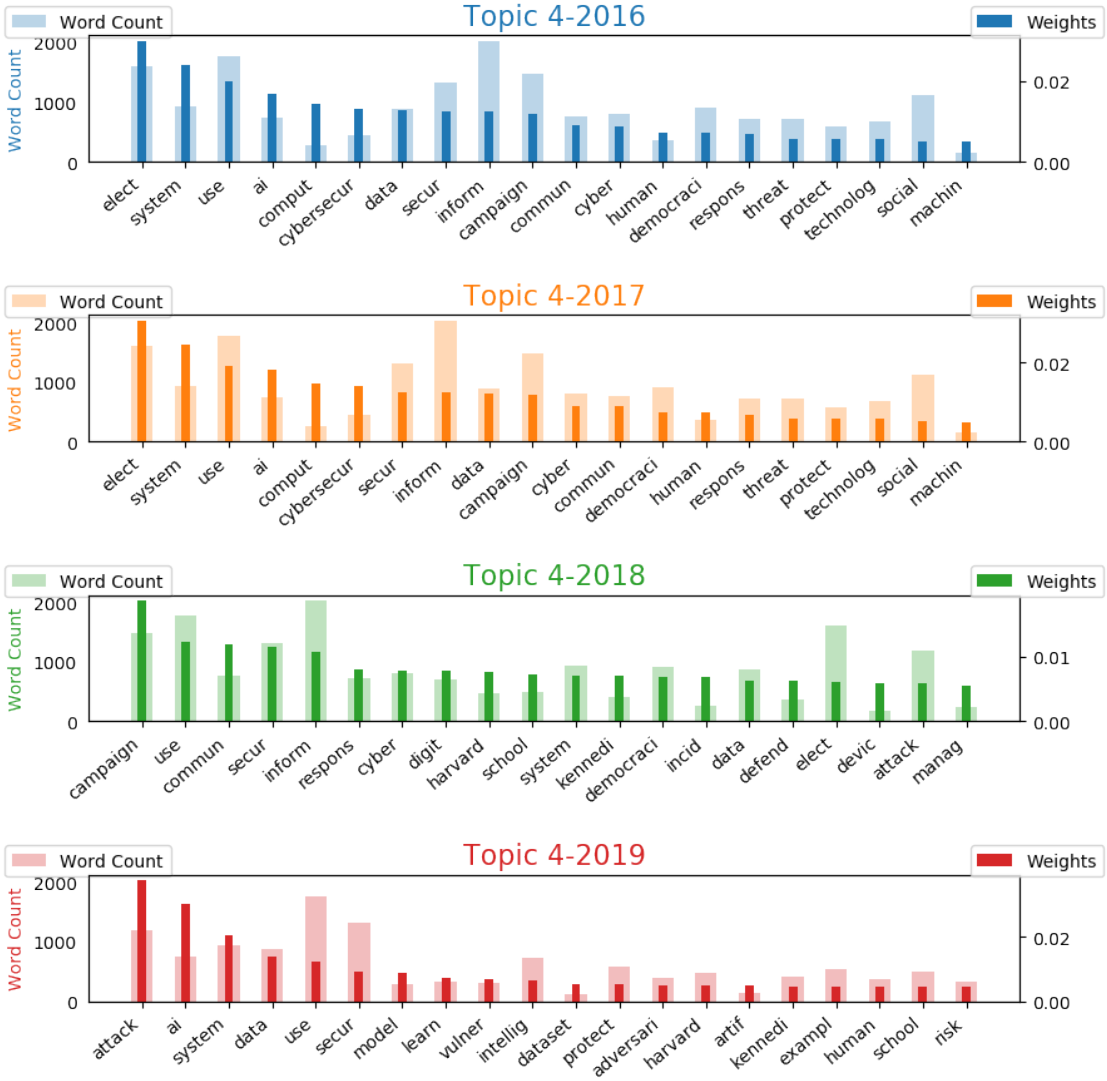
[58]:
```
#the second topic
vis_df_2 = construct_vis_df(processed_df, second_topic_5)
times_2 = ["Topic 2-2016", "Topic 2-2017",
         "Topic 2-2018", "Topic 2-2019"]
graph_topic_keyword_count(vis_df_2, 'Topic2', times_2)
```



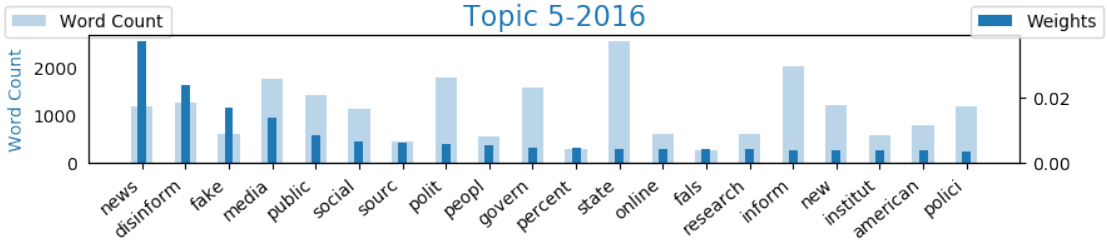
[59]:
```
#the third topic
vis_df_3 = construct_vis_df(processed_df, third_topic_5)
times_3 = ["Topic 3-2016", "Topic 3-2017",
           "Topic 3-2018", "Topic 3-2019"]
graph_topic_keyword_count(vis_df_3, 'Topic3', times_3)
```

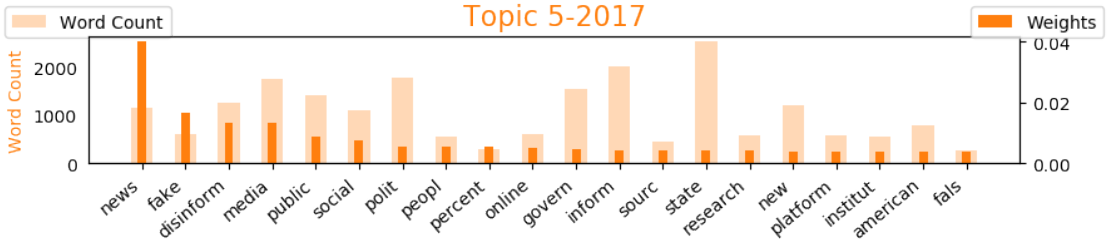


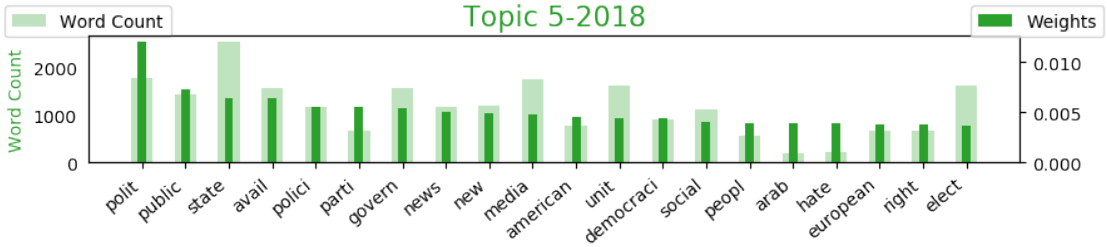
[60]:
```
#the fourth topic
vis_df_4 = construct_vis_df(processed_df, fourth_topic_5)
times_4 = ["Topic 4-2016", "Topic 4-2017",
           "Topic 4-2018", "Topic 4-2019"]
graph_topic_keyword_count(vis_df_4, 'Topic4', times_4)
```

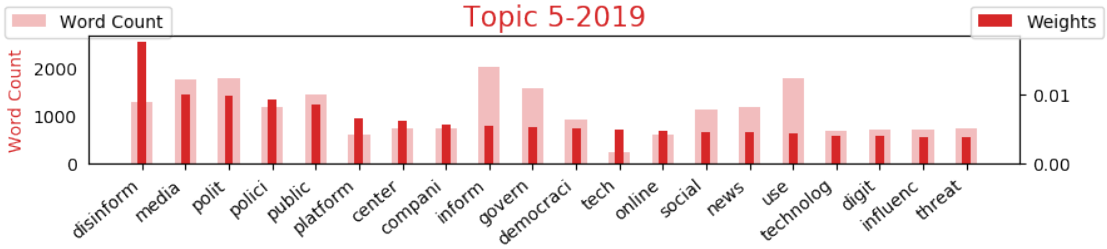


[61]:
```
#the fifth topic
vis_df_5 = construct_vis_df(processed_df, fifth_topic_5)
times_5 = ["Topic 5-2016", "Topic 5-2017",
           "Topic 5-2018", "Topic 5-2019"]
graph_topic_keyword_count(vis_df_5, 'Topic5', times_5)
```









[31]:
```
# fit the 6-topic model
start = time.time()
ldaseq_6 = ldaseqmodel.LdaSeqModel(corpus=corpus_all, id2word=dic_all,␣
 ↪time_slice=time_slice,
                                    num_topics=6, chain_variance=0.1)
end = time.time()
print(end - start)
```

```
/Users/ditong/anaconda3/lib/python3.7/site-
packages/gensim/models/ldaseqmodel.py:230: RuntimeWarning: divide by zero
encountered in double_scalars
  convergence = np.fabs((bound - old_bound) / old_bound)
```

```
2807.1138439178467
```

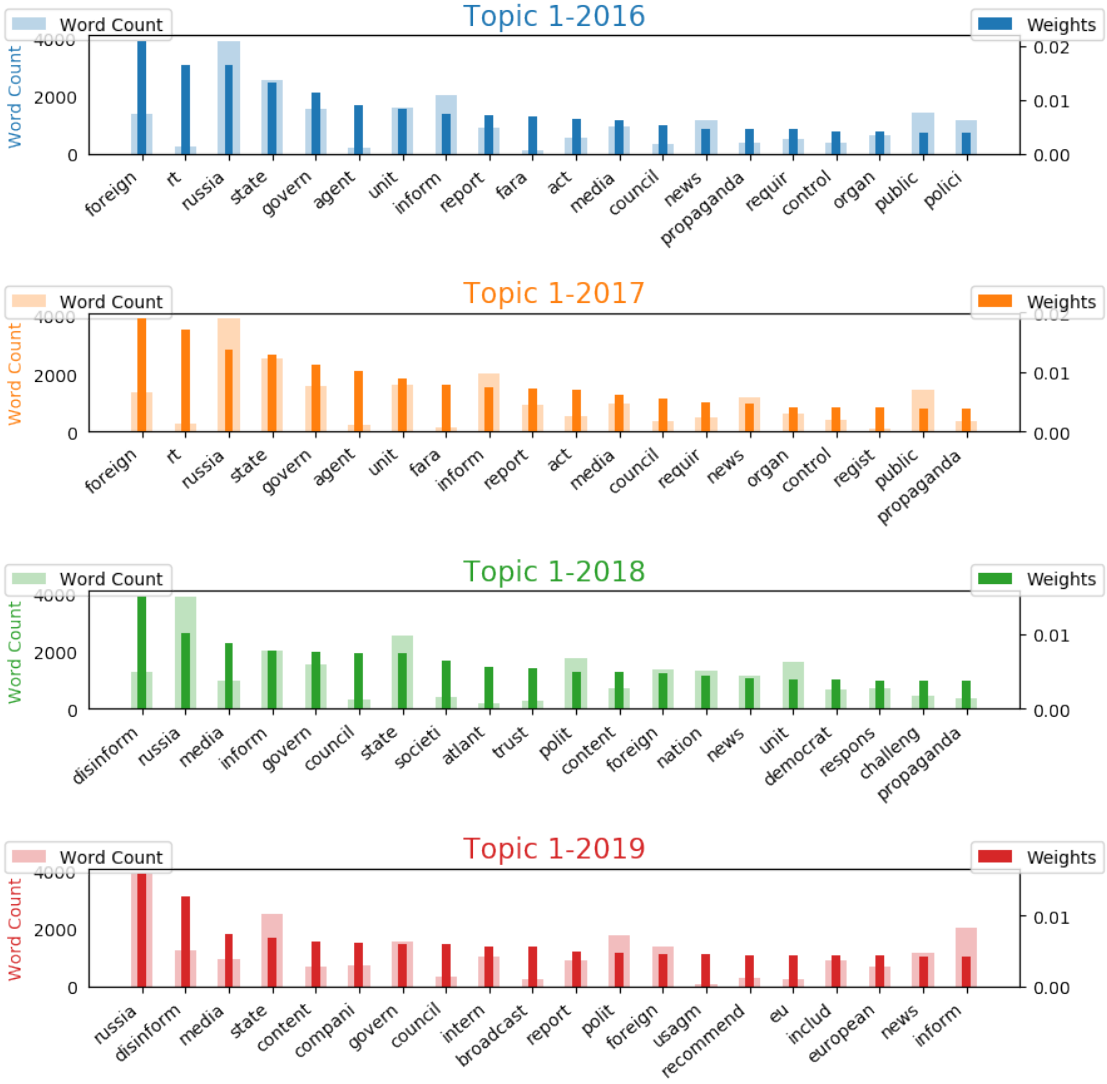
[32]:
```
# save model
pickle.dump(ldaseq_6, open("ldaseq_model_6_final.sav", 'wb'))
```

[33]:
```
# calculate coherence matrix
topics_dtm_61 = ldaseq_6.dtm_coherence(time=0)
topics_dtm_62 = ldaseq_6.dtm_coherence(time=1)
topics_dtm_63 = ldaseq_6.dtm_coherence(time=2)
topics_dtm_64 = ldaseq_6.dtm_coherence(time=3)
cm_DTM_61 = CoherenceModel(topics=topics_dtm_61, corpus=corpus_all,␣
 ↪dictionary=dic_all, coherence='u_mass')
cm_DTM_62 = CoherenceModel(topics=topics_dtm_62, corpus=corpus_all,␣
 ↪dictionary=dic_all, coherence='u_mass')
cm_DTM_63 = CoherenceModel(topics=topics_dtm_63, corpus=corpus_all,␣
 ↪dictionary=dic_all, coherence='u_mass')
cm_DTM_64 = CoherenceModel(topics=topics_dtm_64, corpus=corpus_all,␣
 ↪dictionary=dic_all, coherence='u_mass')
print ("U_mass topic coherence")
print ("DTM Python coherence is", cm_DTM_61.get_coherence(), ";", cm_DTM_62.
 ↪get_coherence(), ";",
      cm_DTM_63.get_coherence(), ";", cm_DTM_64.get_coherence())
```

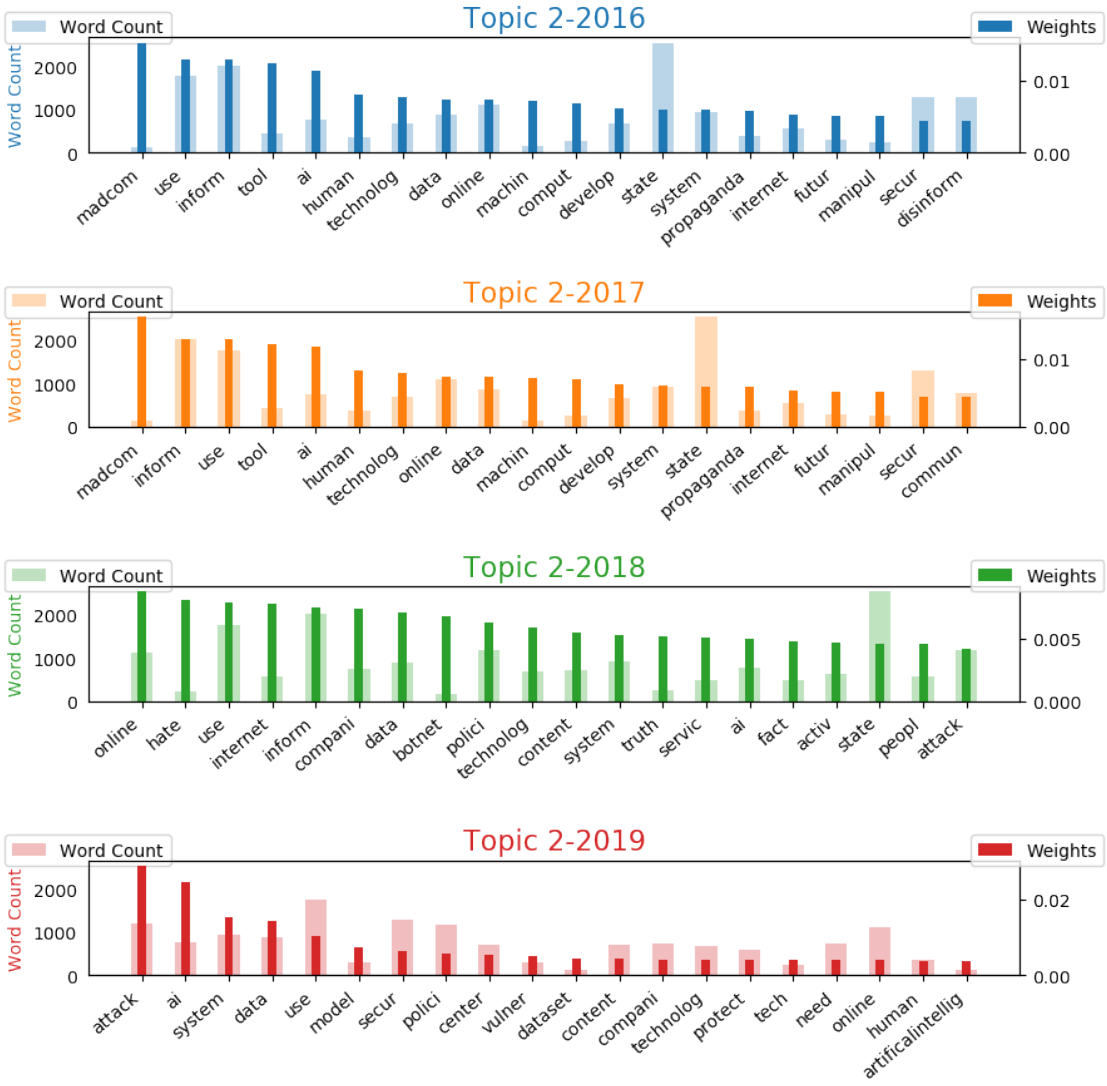
```
U_mass topic coherence
DTM Python coherence is -0.27655205596041954 ; -0.2274416383898592 ;
-0.27139429101928925 ; -0.39355188588791506
```

[34]:
```
# print topic evolution
first_topic_6 = ldaseq_6.print_topic_times(topic=0)
second_topic_6 = ldaseq_6.print_topic_times(topic=1)
third_topic_6 = ldaseq_6.print_topic_times(topic=2)
fourth_topic_6 = ldaseq_6.print_topic_times(topic=3)
fifth_topic_6 = ldaseq_6.print_topic_times(topic=4)
sixth_topic_6 = ldaseq_6.print_topic_times(topic=5)
```

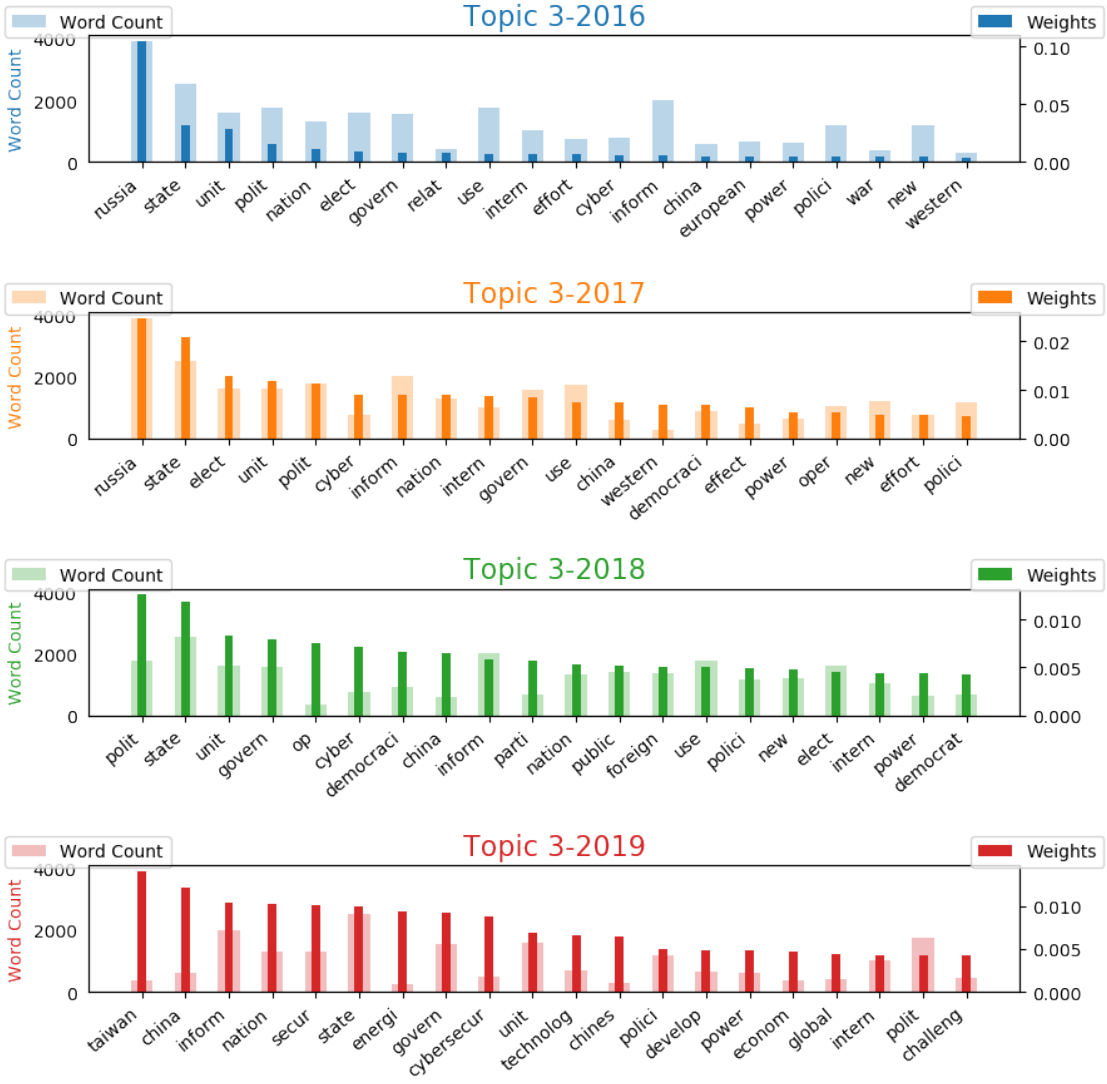
[35]:
```
#the first topic
vis_df_61 = construct_vis_df(processed_df, first_topic_6)
times_61 = ["Topic 1-2016", "Topic 1-2017",
         "Topic 1-2018", "Topic 1-2019"]
graph_topic_keyword_count(vis_df_61, 'Topic1', times_61)
```



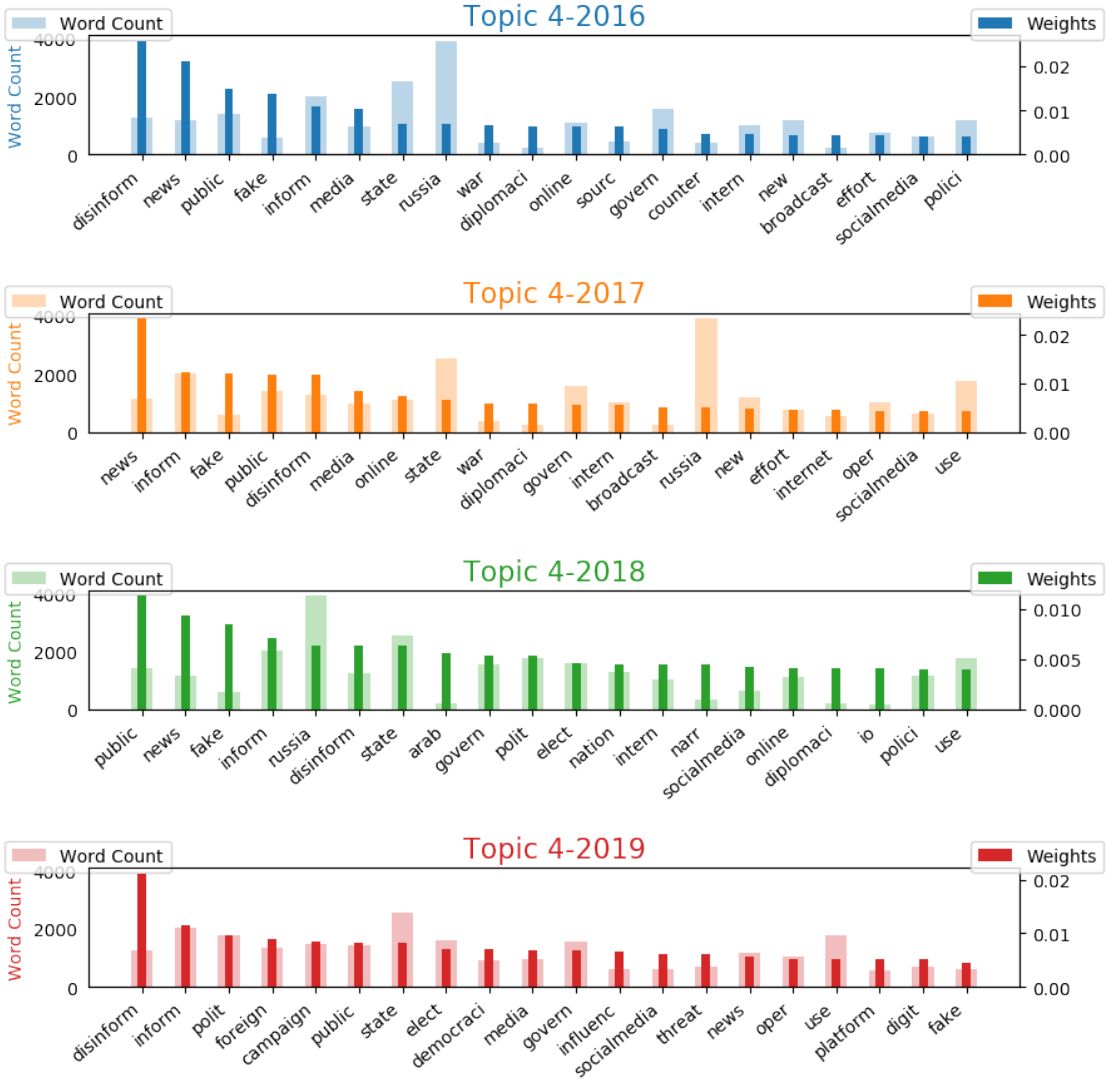
[36]:
```
#the second topic
vis_df_62 = construct_vis_df(processed_df, second_topic_6)
times_62 = ["Topic 2-2016", "Topic 2-2017",
         "Topic 2-2018", "Topic 2-2019"]
graph_topic_keyword_count(vis_df_62, 'Topic2', times_62)
```



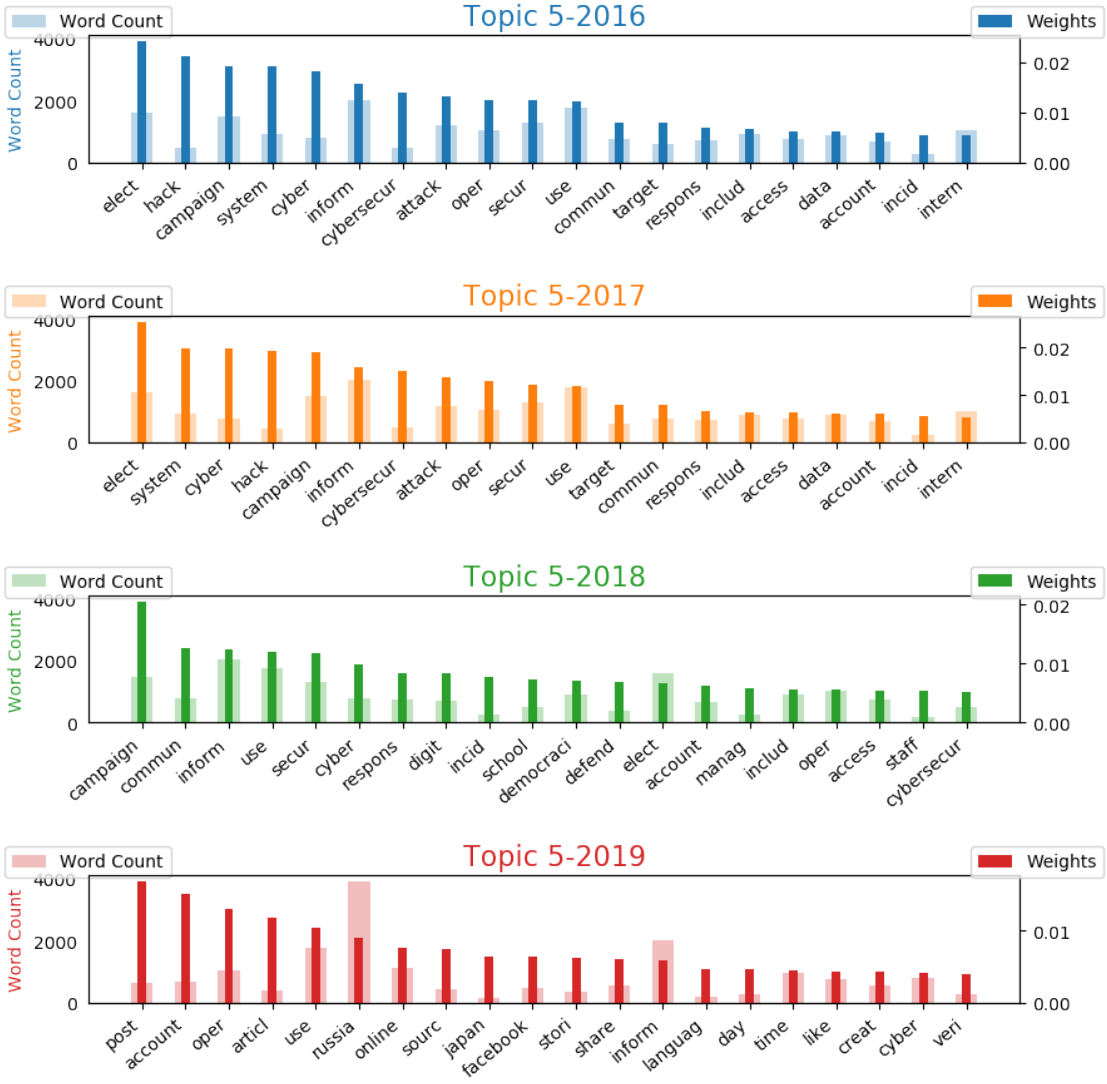
[37]:
```
#the third topic
vis_df_63 = construct_vis_df(processed_df, third_topic_6)
times_63 = ["Topic 3-2016", "Topic 3-2017",
         "Topic 3-2018", "Topic 3-2019"]
graph_topic_keyword_count(vis_df_63, 'Topic3', times_63)
```



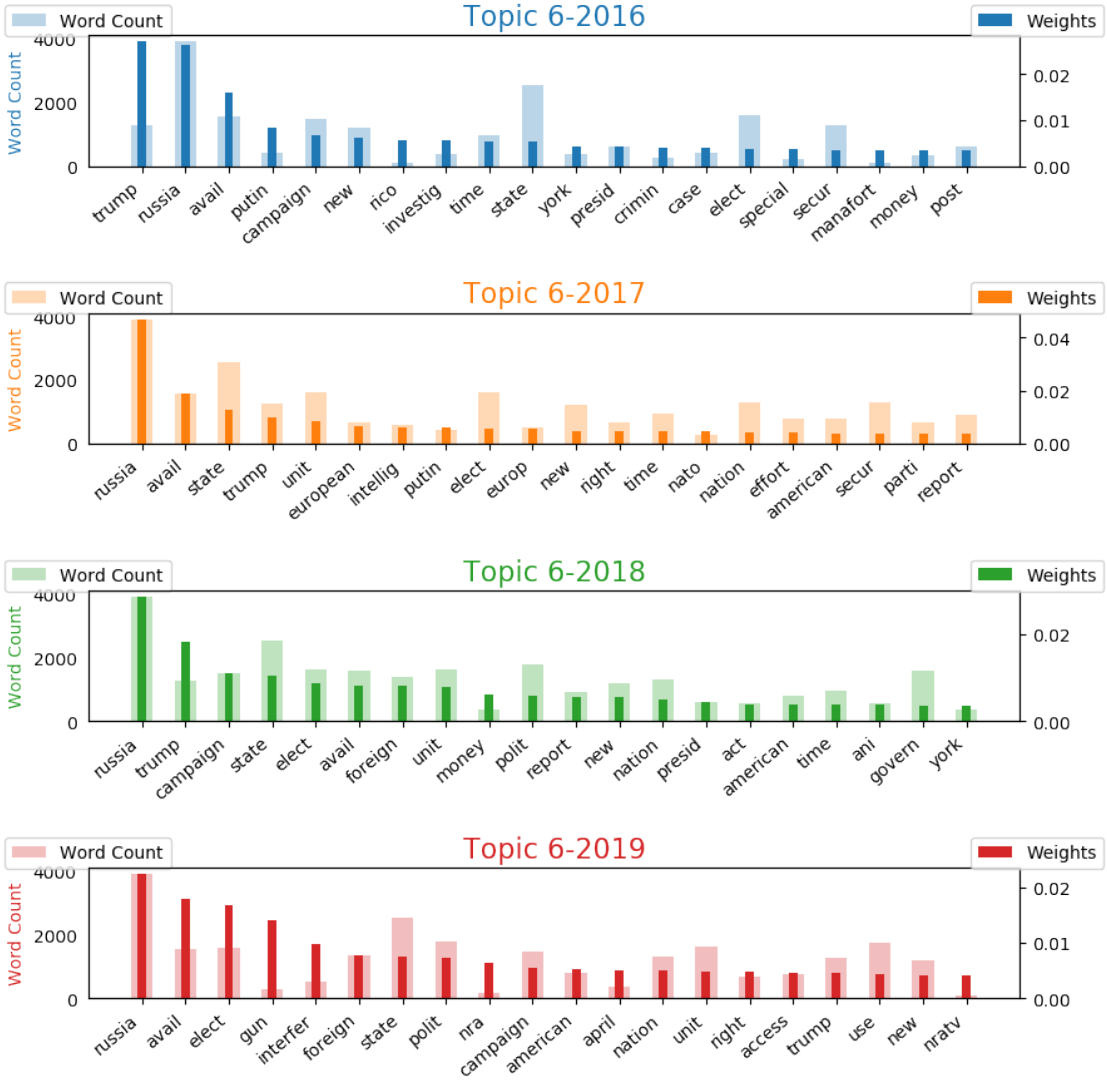
[38]:
```
#the fourth topic
vis_df_64 = construct_vis_df(processed_df, fourth_topic_6)
times_64 = ["Topic 4-2016", "Topic 4-2017",
         "Topic 4-2018", "Topic 4-2019"]
graph_topic_keyword_count(vis_df_64, 'Topic4', times_64)
```



[39]:
```
#the fifth topic
vis_df_65 = construct_vis_df(processed_df, fifth_topic_6)
times_65 = ["Topic 5-2016", "Topic 5-2017",
         "Topic 5-2018", "Topic 5-2019"]
graph_topic_keyword_count(vis_df_65, 'Topic5', times_65)
```



[40]:
```
#the sixth topic
vis_df_66 = construct_vis_df(processed_df, sixth_topic_6)
times_66 = ["Topic 6-2016", "Topic 6-2017",
         "Topic 6-2018", "Topic 6-2019"]
graph_topic_keyword_count(vis_df_66, 'Topic6', times_66)
```



[92]:
```
# fit the 7-topic model
start = time.time()
ldaseq_7 = ldaseqmodel.LdaSeqModel(corpus=corpus_all, id2word=dic_all,␣
 ↪time_slice=time_slice,
                                    num_topics=7, chain_variance=0.1)
end = time.time()
print(end - start)
```

```
/Users/ditong/anaconda3/lib/python3.7/site-
packages/gensim/models/ldaseqmodel.py:230: RuntimeWarning: divide by zero
encountered in double_scalars
  convergence = np.fabs((bound - old_bound) / old_bound)
```

```
3084.260115146637
```

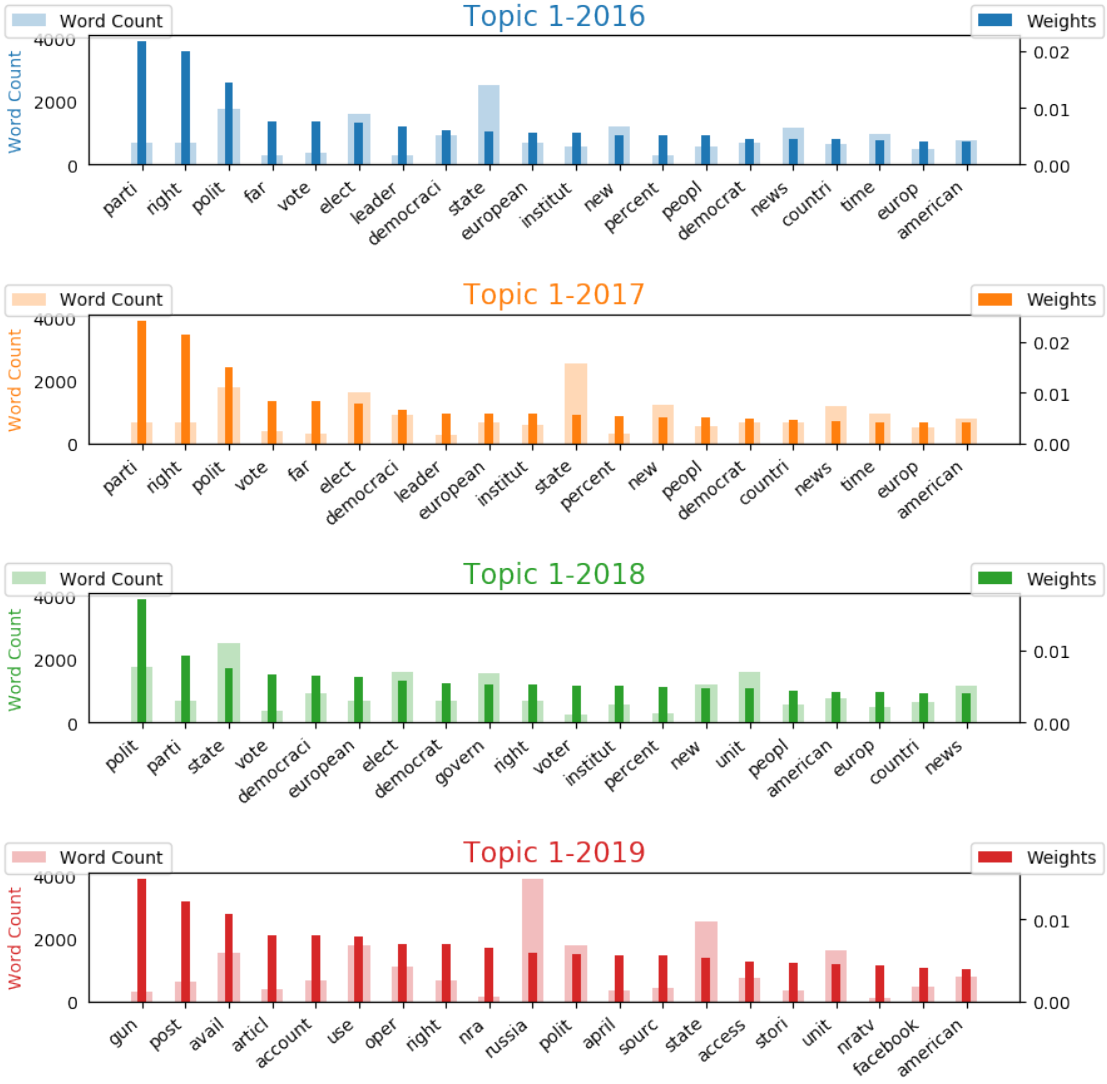
[93]:
```
# save model
pickle.dump(ldaseq_7, open("ldaseq_model_7_new.sav", 'wb'))
```

[94]:
```
# calculate coherence matrix
topics_dtm_71 = ldaseq_7.dtm_coherence(time=0)
topics_dtm_72 = ldaseq_7.dtm_coherence(time=1)
topics_dtm_73 = ldaseq_7.dtm_coherence(time=2)
topics_dtm_74 = ldaseq_7.dtm_coherence(time=3)
cm_DTM_71 = CoherenceModel(topics=topics_dtm_71, corpus=corpus_all,␣
 ↪dictionary=dic_all, coherence='u_mass')
cm_DTM_72 = CoherenceModel(topics=topics_dtm_72, corpus=corpus_all,␣
 ↪dictionary=dic_all, coherence='u_mass')
cm_DTM_73 = CoherenceModel(topics=topics_dtm_73, corpus=corpus_all,␣
 ↪dictionary=dic_all, coherence='u_mass')
cm_DTM_74 = CoherenceModel(topics=topics_dtm_74, corpus=corpus_all,␣
 ↪dictionary=dic_all, coherence='u_mass')
print ("U_mass topic coherence")
print ("DTM Python coherence is", cm_DTM_71.get_coherence(), ";", cm_DTM_72.
 ↪get_coherence(), ";",
      cm_DTM_73.get_coherence(), ";", cm_DTM_74.get_coherence())
```

```
U_mass topic coherence
DTM Python coherence is -0.2561849350494962 ; -0.22734723658233744 ;
-0.2820903647048307 ; -0.37988481775889005
```

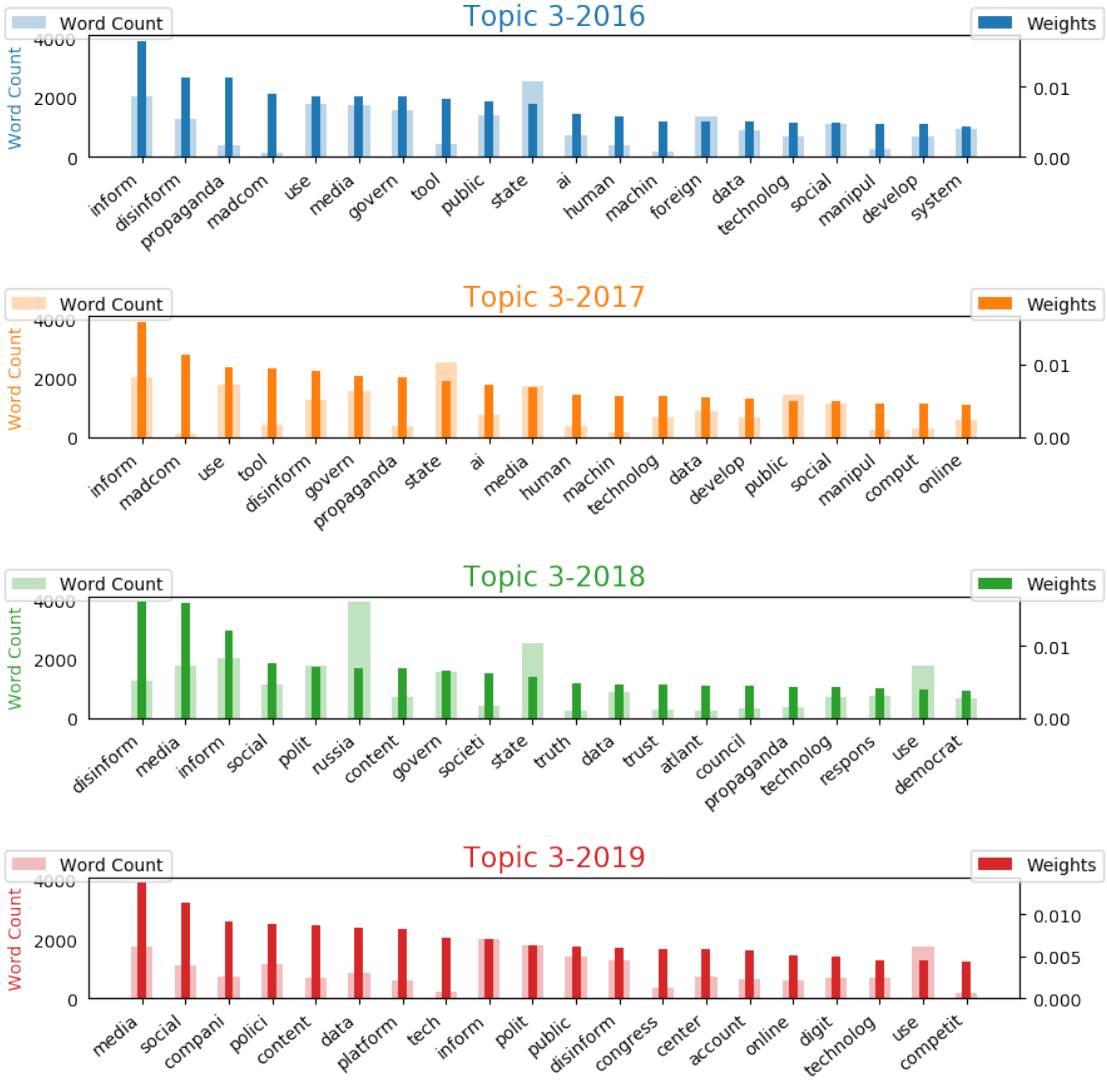
[95]:
```
# print topic evolution
first_topic_7 = ldaseq_7.print_topic_times(topic=0)
second_topic_7 = ldaseq_7.print_topic_times(topic=1)
third_topic_7 = ldaseq_7.print_topic_times(topic=2)
fourth_topic_7 = ldaseq_7.print_topic_times(topic=3)
fifth_topic_7 = ldaseq_7.print_topic_times(topic=4)
sixth_topic_7 = ldaseq_7.print_topic_times(topic=5)
seventh_topic_7 = ldaseq_7.print_topic_times(topic=6)
```

[96]:
```
#the first topic
vis_df_71 = construct_vis_df(processed_df, first_topic_7)
times_71 = ["Topic 1-2016", "Topic 1-2017",
         "Topic 1-2018", "Topic 1-2019"]
graph_topic_keyword_count(vis_df_71, 'Topic1', times_71)
```

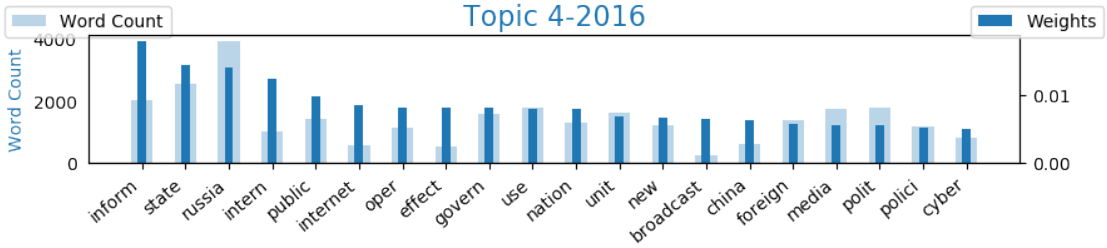


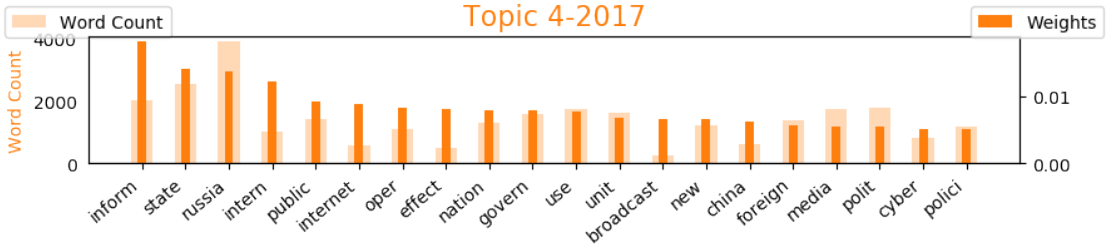
[97]:
```
#the second topic
vis_df_72 = construct_vis_df(processed_df, second_topic_7)
times_72 = ["Topic 2-2016", "Topic 2-2017",
         "Topic 2-2018", "Topic 2-2019"]
graph_topic_keyword_count(vis_df_72, 'Topic2', times_72)
```

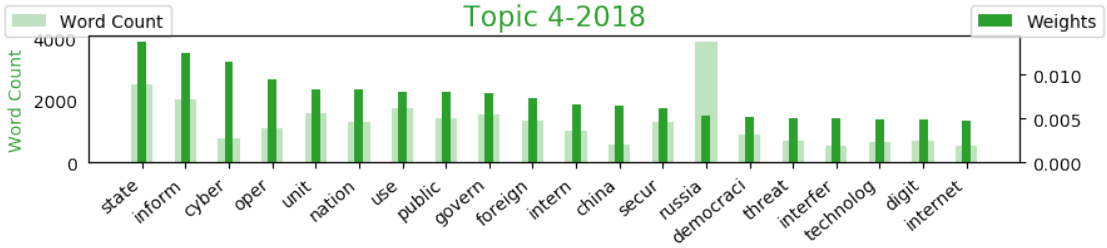
[98]:
```
#the third topic
vis_df_73 = construct_vis_df(processed_df, third_topic_7)
times_73 = ["Topic 3-2016", "Topic 3-2017",
         "Topic 3-2018", "Topic 3-2019"]
graph_topic_keyword_count(vis_df_73, 'Topic3', times_73)
```

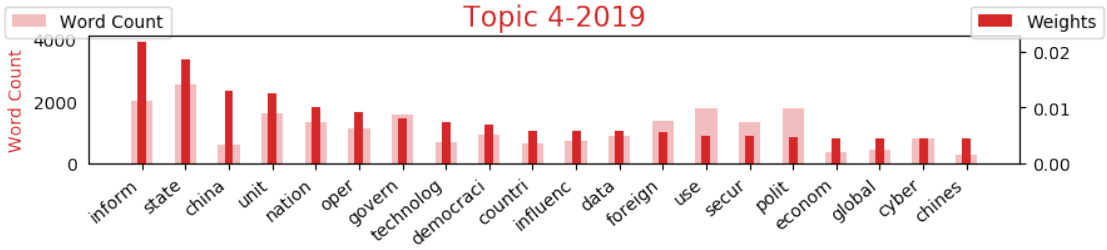


[99]:
```
#the fourth topic
vis_df_74 = construct_vis_df(processed_df, fourth_topic_7)
times_74 = ["Topic 4-2016", "Topic 4-2017",
         "Topic 4-2018", "Topic 4-2019"]
graph_topic_keyword_count(vis_df_74, 'Topic4', times_74)
```

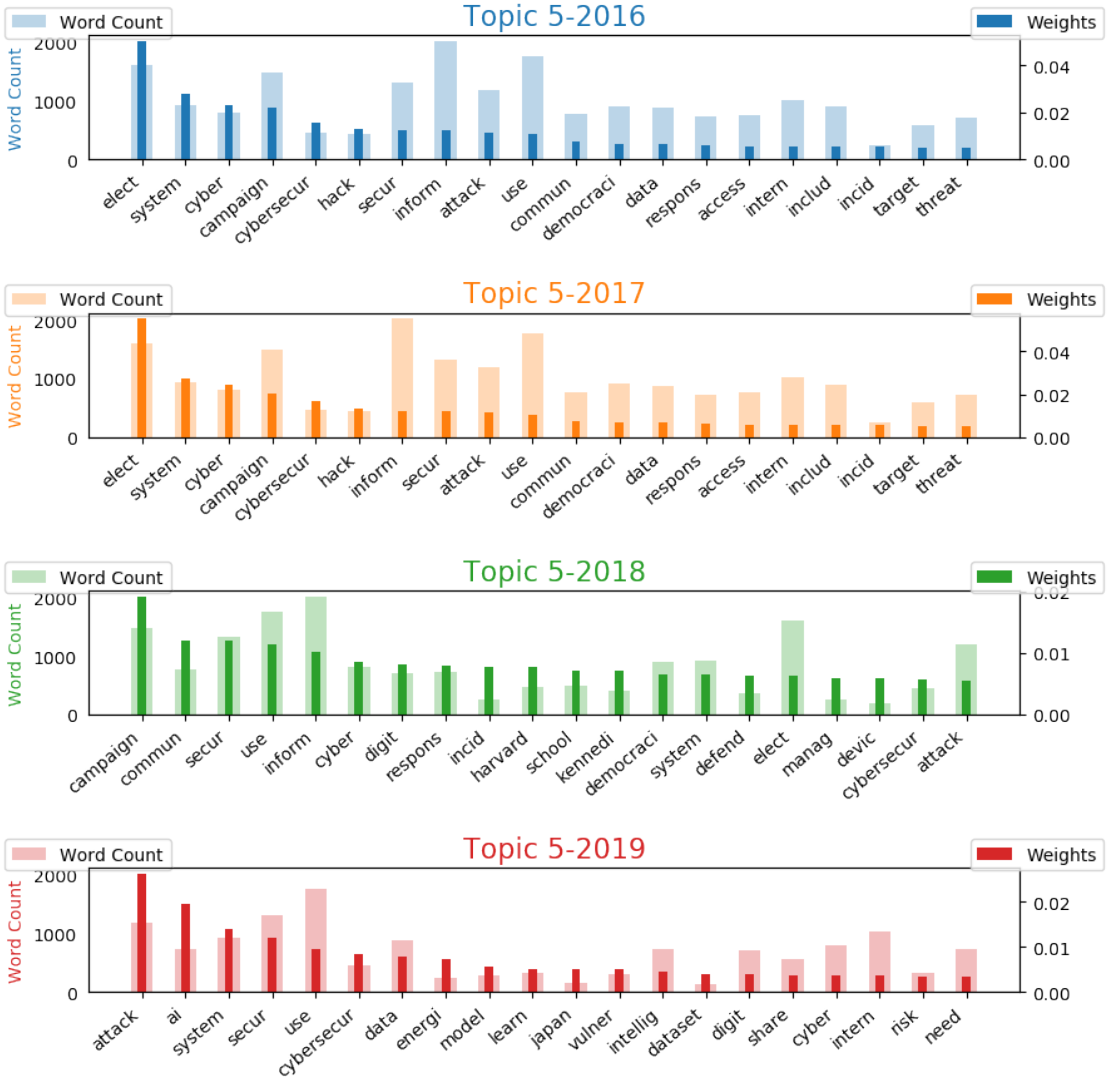




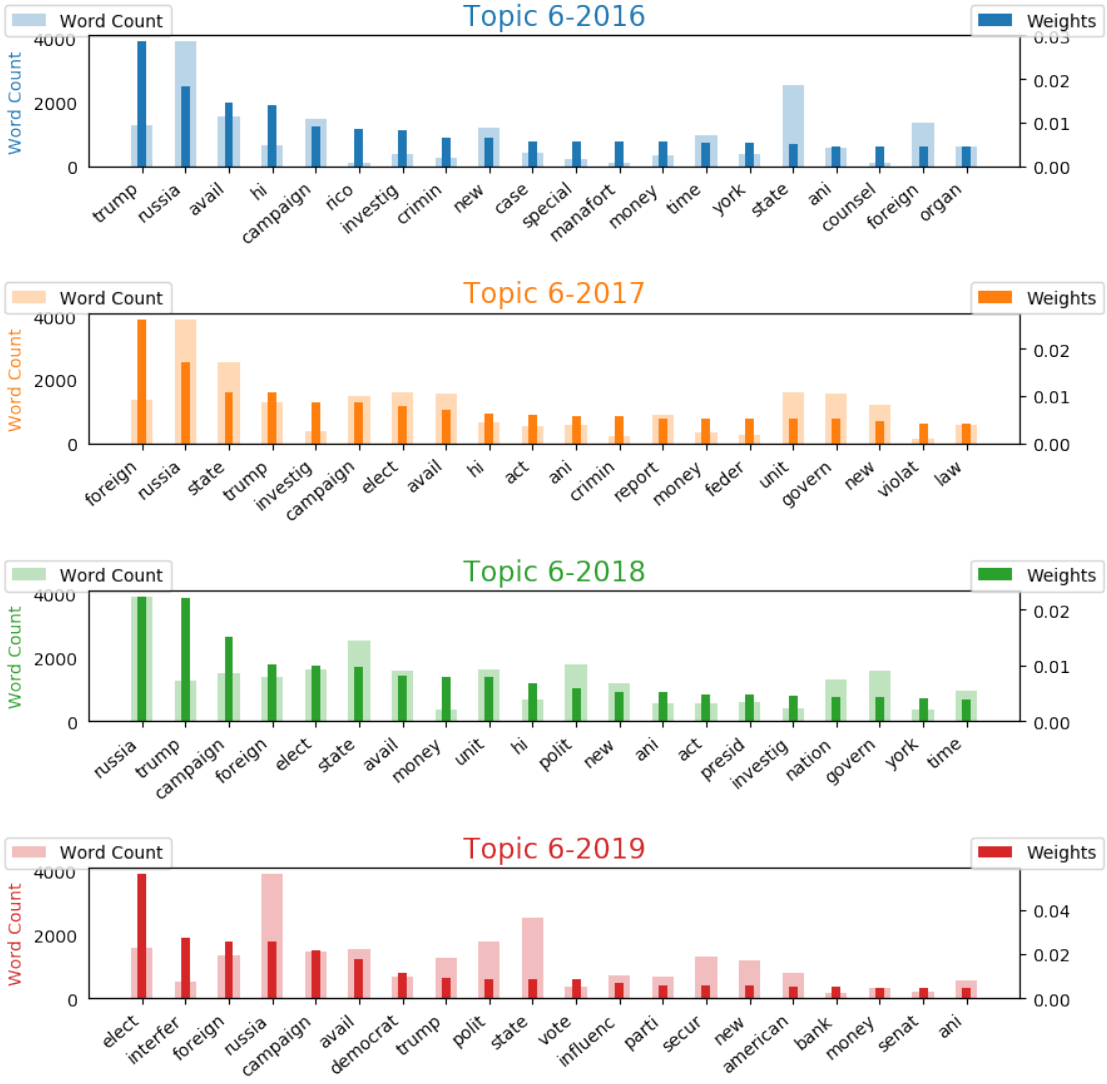




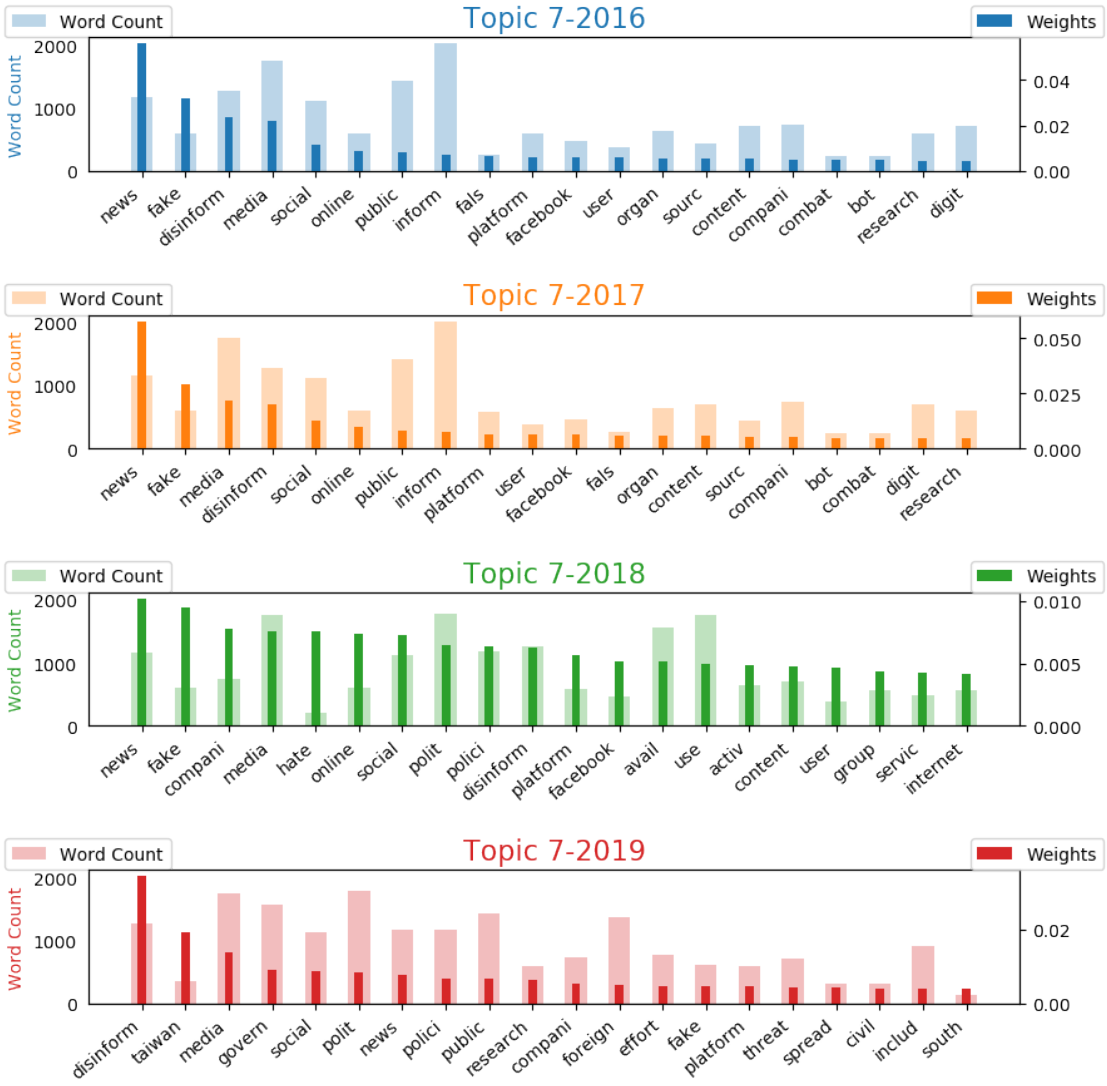
[100]:
```
#the fifth topic
vis_df_75 = construct_vis_df(processed_df, fifth_topic_7)
times_75 = ["Topic 5-2016", "Topic 5-2017",
         "Topic 5-2018", "Topic 5-2019"]
graph_topic_keyword_count(vis_df_75, 'Topic5', times_75)
```



[101]:
```
#the sixth topic
vis_df_76 = construct_vis_df(processed_df, sixth_topic_7)
times_76 = ["Topic 6-2016", "Topic 6-2017",
         "Topic 6-2018", "Topic 6-2019"]
graph_topic_keyword_count(vis_df_76, 'Topic6', times_76)
```



[102]:
```
#the seventh topic
vis_df_77 = construct_vis_df(processed_df, seventh_topic_7)
times_77 = ["Topic 7-2016", "Topic 7-2017",
         "Topic 7-2018", "Topic 7-2019"]
graph_topic_keyword_count(vis_df_77, 'Topic7', times_77)
```



[120]:
```
# fit the 8-topic model
start = time.time()
ldaseq_8 = ldaseqmodel.LdaSeqModel(corpus=corpus_all, id2word=dic_all,␣
 ↪time_slice=time_slice,
                                    num_topics=8, chain_variance=0.1)
end = time.time()
print(end - start)
```

```
/Users/ditong/anaconda3/lib/python3.7/site-
packages/gensim/models/ldaseqmodel.py:230: RuntimeWarning: divide by zero
encountered in double_scalars
  convergence = np.fabs((bound - old_bound) / old_bound)
```

```
3452.9803540706635
```

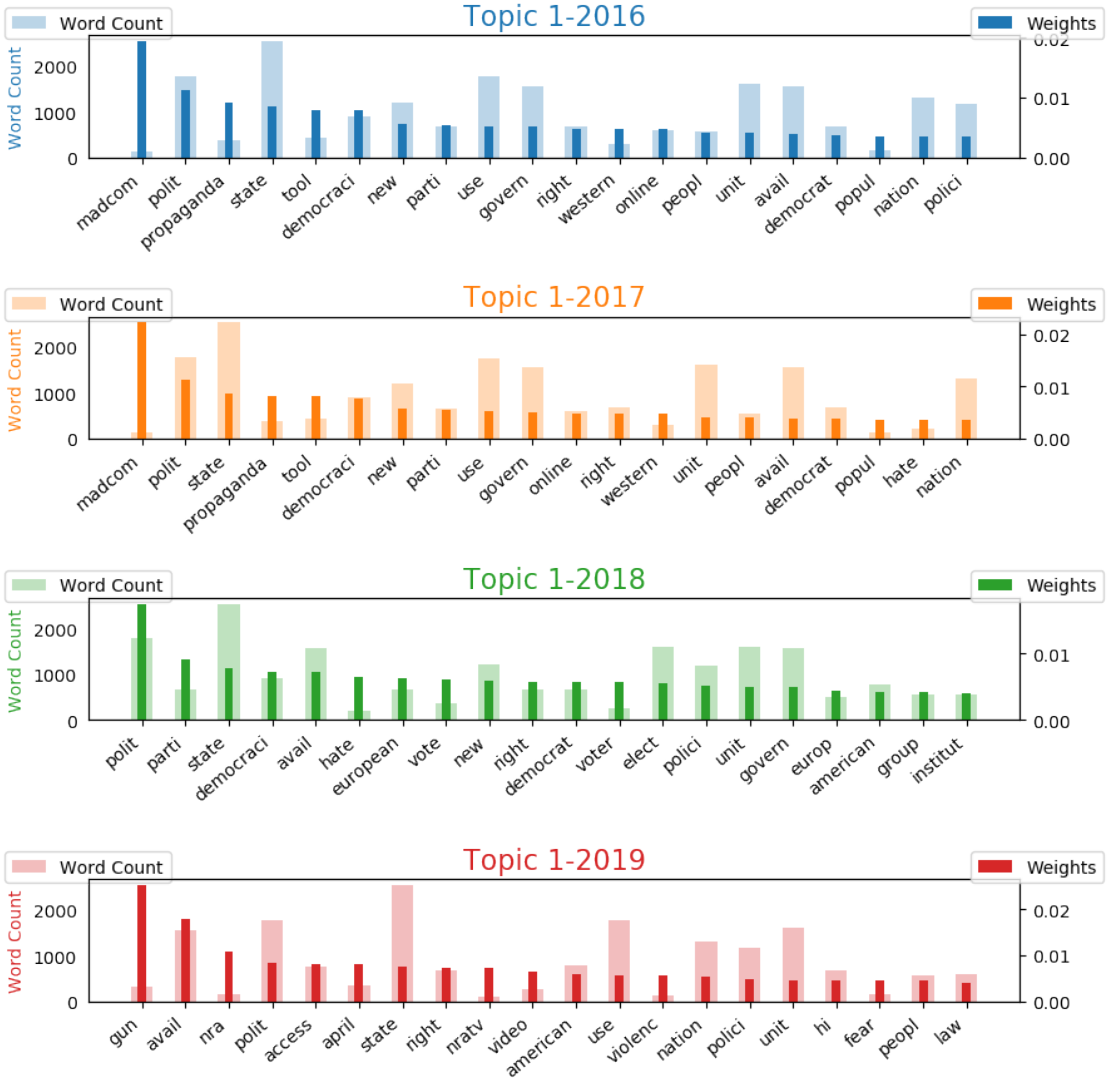
[121]:
```
# save model
pickle.dump(ldaseq_8, open("ldaseq_model_8_new.sav", 'wb'))
```

[122]:
```
# calculate coherence matrix
topics_dtm_81 = ldaseq_8.dtm_coherence(time=0)
topics_dtm_82 = ldaseq_8.dtm_coherence(time=1)
topics_dtm_83 = ldaseq_8.dtm_coherence(time=2)
topics_dtm_84 = ldaseq_8.dtm_coherence(time=3)
cm_DTM_81 = CoherenceModel(topics=topics_dtm_81, corpus=corpus_all,␣
 ↪dictionary=dic_all, coherence='u_mass')
cm_DTM_82 = CoherenceModel(topics=topics_dtm_82, corpus=corpus_all,␣
 ↪dictionary=dic_all, coherence='u_mass')
cm_DTM_83 = CoherenceModel(topics=topics_dtm_83, corpus=corpus_all,␣
 ↪dictionary=dic_all, coherence='u_mass')
cm_DTM_84 = CoherenceModel(topics=topics_dtm_84, corpus=corpus_all,␣
 ↪dictionary=dic_all, coherence='u_mass')
print ("U_mass topic coherence")
print ("DTM Python coherence is", cm_DTM_81.get_coherence(), ";", cm_DTM_82.
 ↪get_coherence(), ";",
      cm_DTM_83.get_coherence(), ";", cm_DTM_84.get_coherence())
```

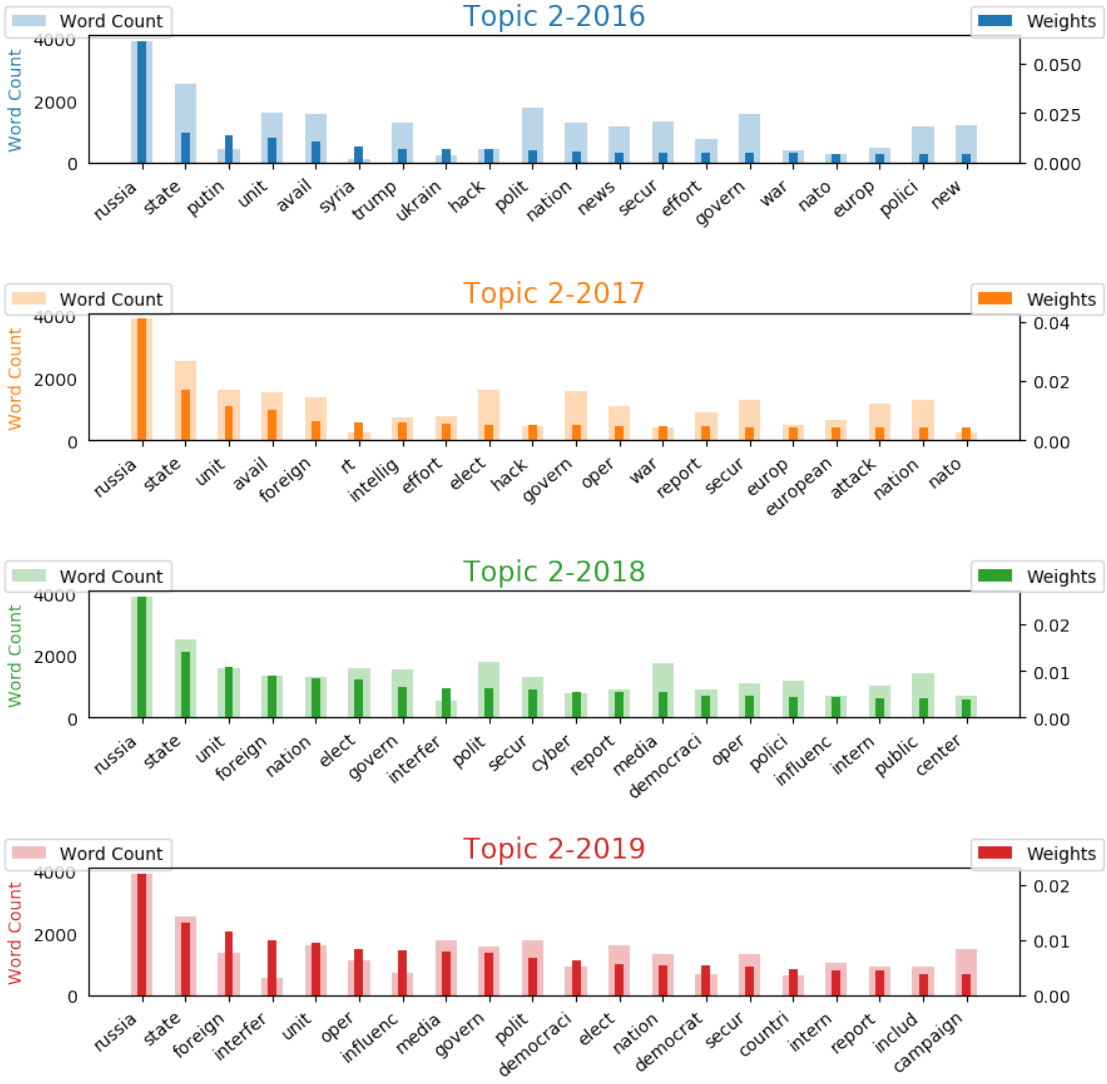
```
U_mass topic coherence
DTM Python coherence is -0.30878740285226564 ; -0.27543444032114517 ;
-0.28487040187887847 ; -0.27201221935180286
```

[123]:
```
# print topic evolution
first_topic_8 = ldaseq_8.print_topic_times(topic=0)
second_topic_8 = ldaseq_8.print_topic_times(topic=1)
third_topic_8 = ldaseq_8.print_topic_times(topic=2)
fourth_topic_8 = ldaseq_8.print_topic_times(topic=3)
fifth_topic_8 = ldaseq_8.print_topic_times(topic=4)
sixth_topic_8 = ldaseq_8.print_topic_times(topic=5)
seventh_topic_8 = ldaseq_8.print_topic_times(topic=6)
eighth_topic_8 = ldaseq_8.print_topic_times(topic=7)
```

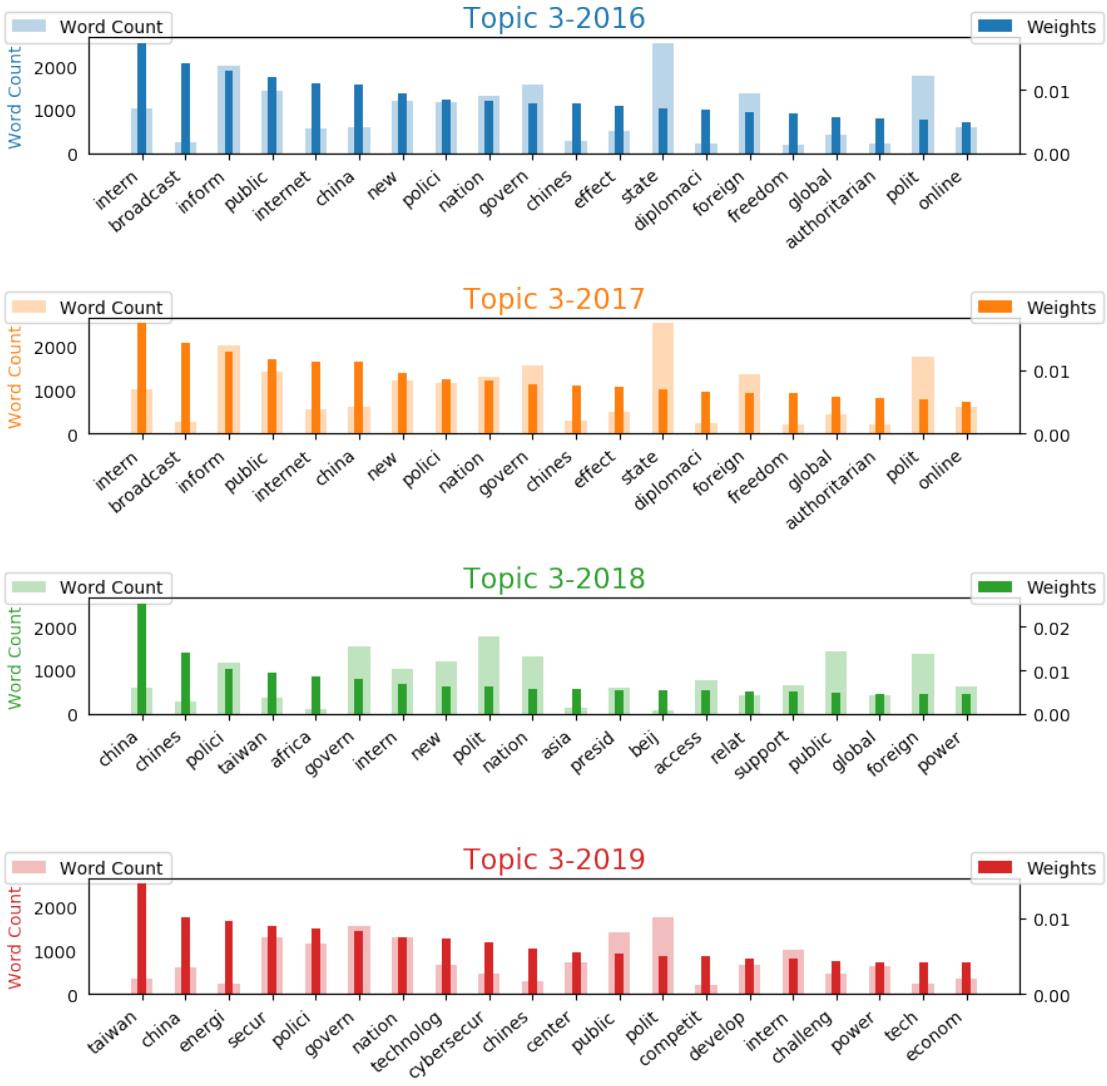
[124]:
```
#the first topic
vis_df_81 = construct_vis_df(processed_df, first_topic_8)
times_81 = ["Topic 1-2016", "Topic 1-2017",
         "Topic 1-2018", "Topic 1-2019"]
graph_topic_keyword_count(vis_df_81, 'Topic1', times_81)
```



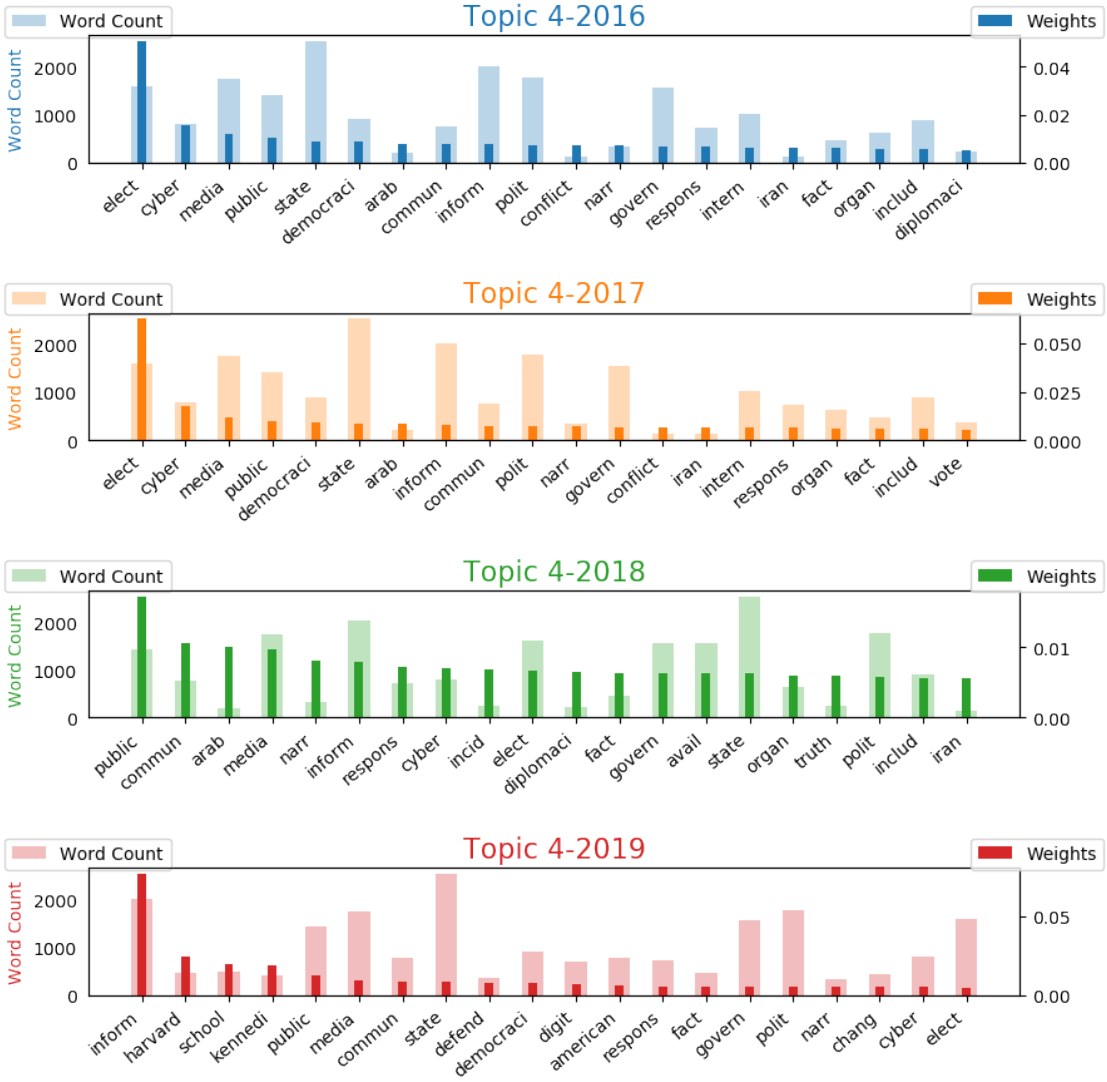
[125]:
```
#the second topic
vis_df_82 = construct_vis_df(processed_df, second_topic_8)
times_82 = ["Topic 2-2016", "Topic 2-2017",
         "Topic 2-2018", "Topic 2-2019"]
graph_topic_keyword_count(vis_df_82, 'Topic2', times_82)
```



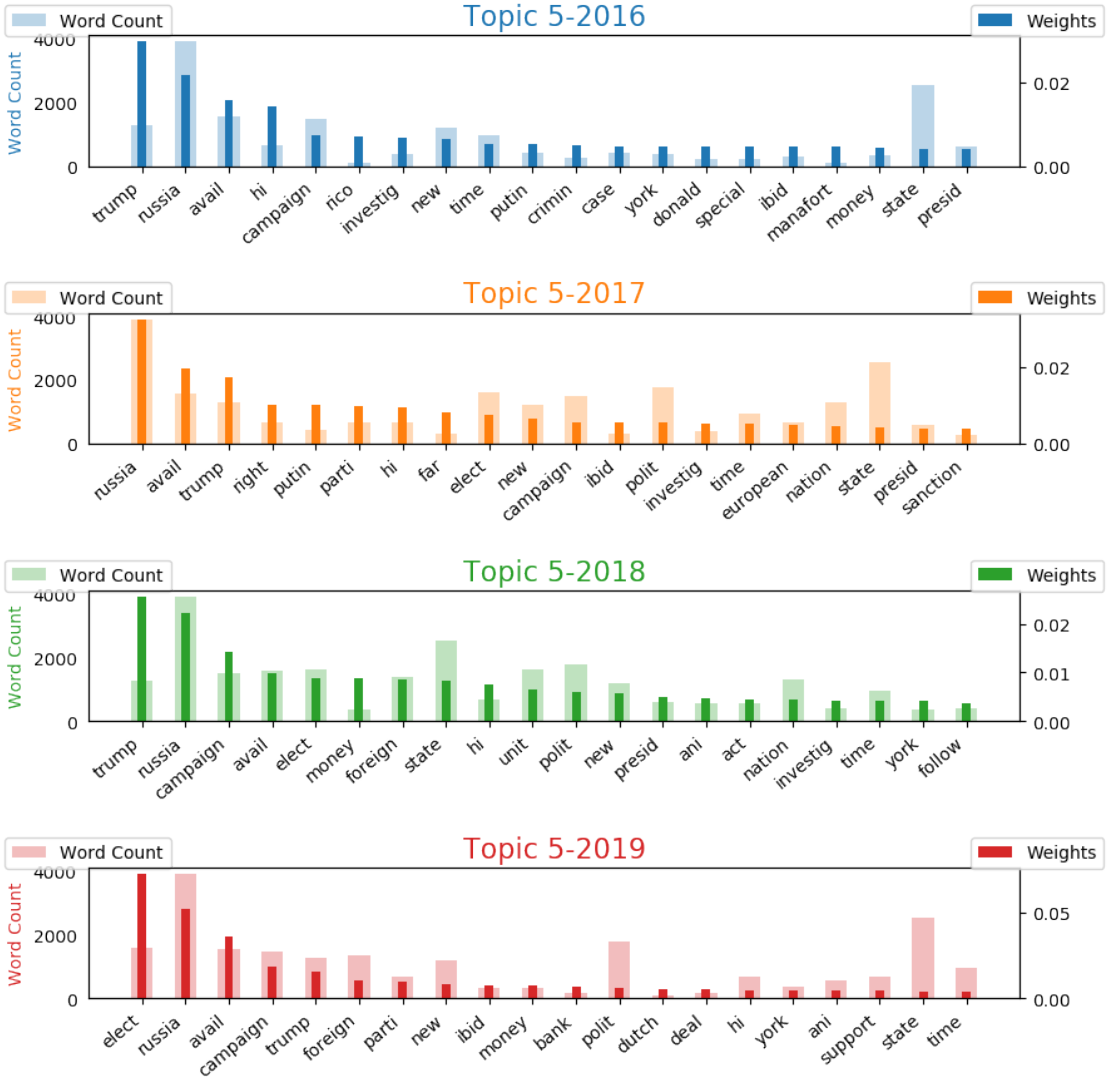
[126]:
```
#the third topic
vis_df_83 = construct_vis_df(processed_df, third_topic_8)
times_83 = ["Topic 3-2016", "Topic 3-2017",
         "Topic 3-2018", "Topic 3-2019"]
graph_topic_keyword_count(vis_df_83, 'Topic3', times_83)
```



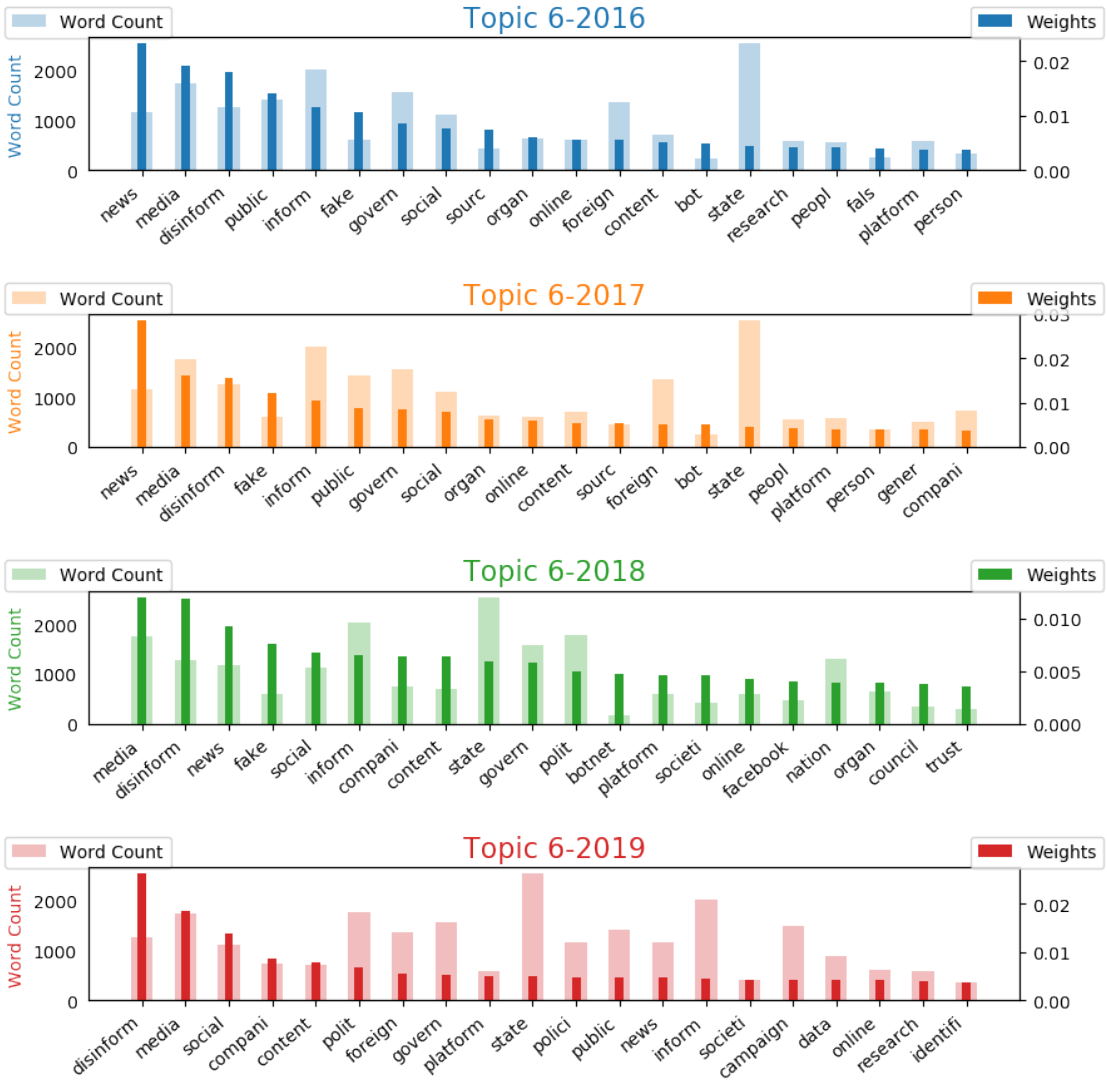
[127]:
```
#the fourth topic
vis_df_84 = construct_vis_df(processed_df, fourth_topic_8)
times_84 = ["Topic 4-2016", "Topic 4-2017",
         "Topic 4-2018", "Topic 4-2019"]
graph_topic_keyword_count(vis_df_84, 'Topic4', times_84)
```



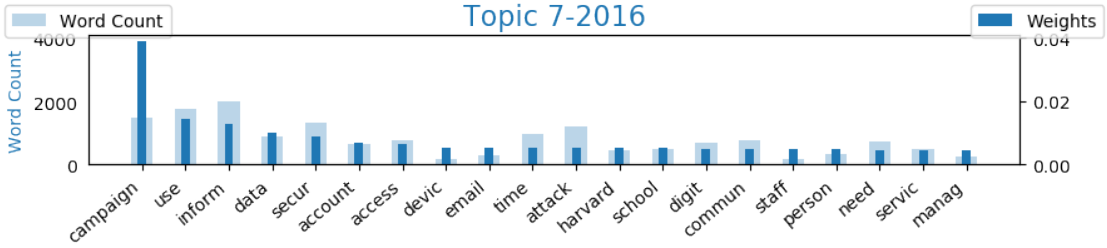
[128]:
```
#the fifth topic
vis_df_85 = construct_vis_df(processed_df, fifth_topic_8)
times_85 = ["Topic 5-2016", "Topic 5-2017",
         "Topic 5-2018", "Topic 5-2019"]
graph_topic_keyword_count(vis_df_85, 'Topic5', times_85)
```

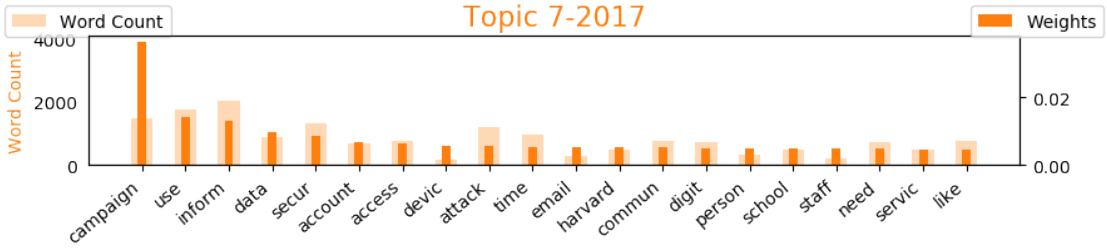


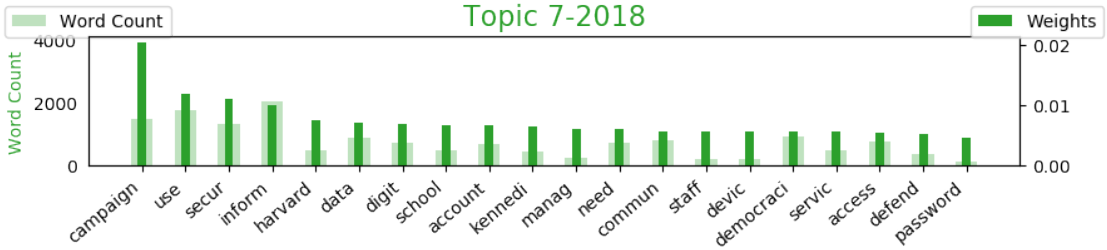
[129]:
```
#the sixth topic
vis_df_86 = construct_vis_df(processed_df, sixth_topic_8)
times_86 = ["Topic 6-2016", "Topic 6-2017",
         "Topic 6-2018", "Topic 6-2019"]
graph_topic_keyword_count(vis_df_86, 'Topic6', times_86)
```

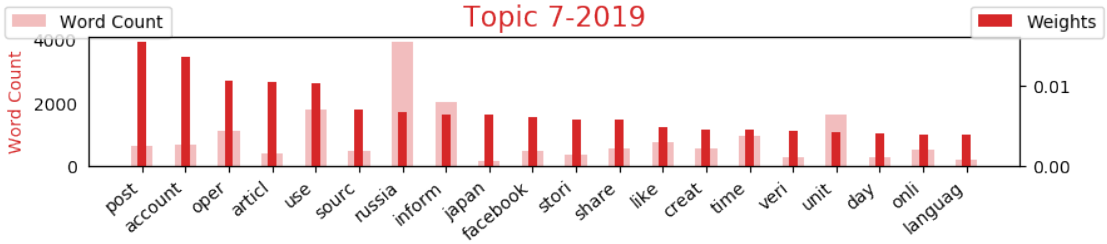


[130]:
```
#the seventh topic
vis_df_87 = construct_vis_df(processed_df, seventh_topic_8)
times_87 = ["Topic 7-2016", "Topic 7-2017",
         "Topic 7-2018", "Topic 7-2019"]
graph_topic_keyword_count(vis_df_87, 'Topic7', times_87)
```

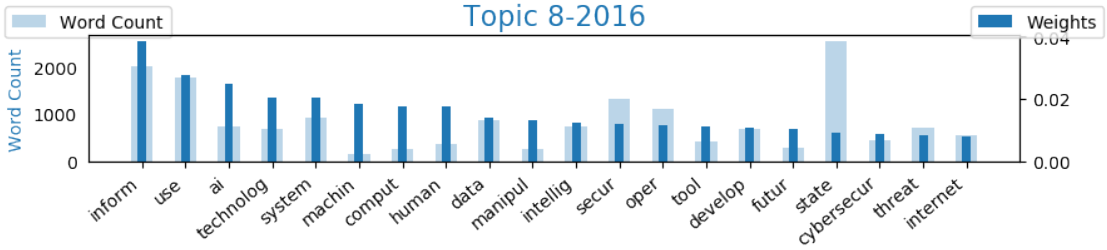


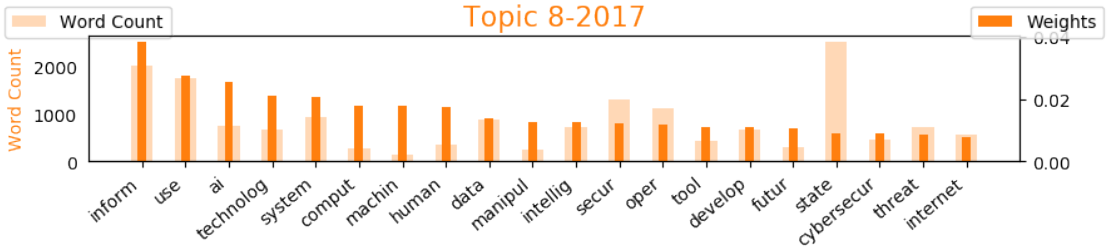


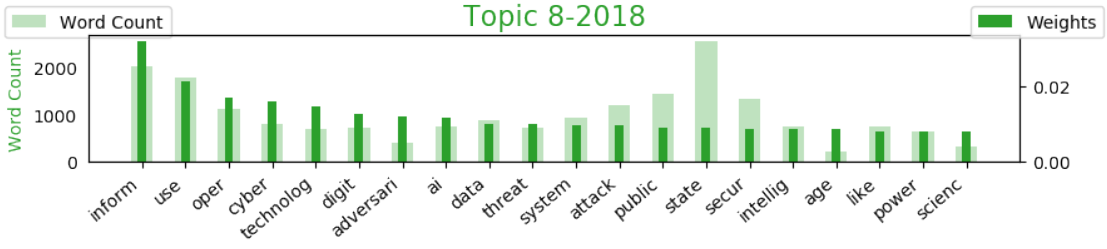


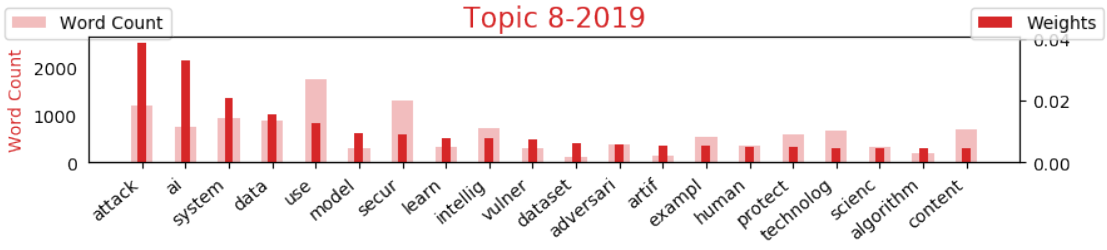


[131]:
```
#the eighth topic
vis_df_88 = construct_vis_df(processed_df, eighth_topic_8)
times_88 = ["Topic 8-2016", "Topic 8-2017",
         "Topic 8-2018", "Topic 8-2019"]
graph_topic_keyword_count(vis_df_88, 'Topic8', times_88)
```









[146]:
```
# fit the 10-topic model
start = time.time()
ldaseq_10 = ldaseqmodel.LdaSeqModel(corpus=corpus_all, id2word=dic_all,␣
 ↪time_slice=time_slice,
                                    num_topics=10, chain_variance=0.1)
end = time.time()
print(end - start)
```

```
/Users/ditong/anaconda3/lib/python3.7/site-
packages/gensim/models/ldaseqmodel.py:230: RuntimeWarning: divide by zero
encountered in double_scalars
  convergence = np.fabs((bound - old_bound) / old_bound)
```

```
3973.3293633461
```

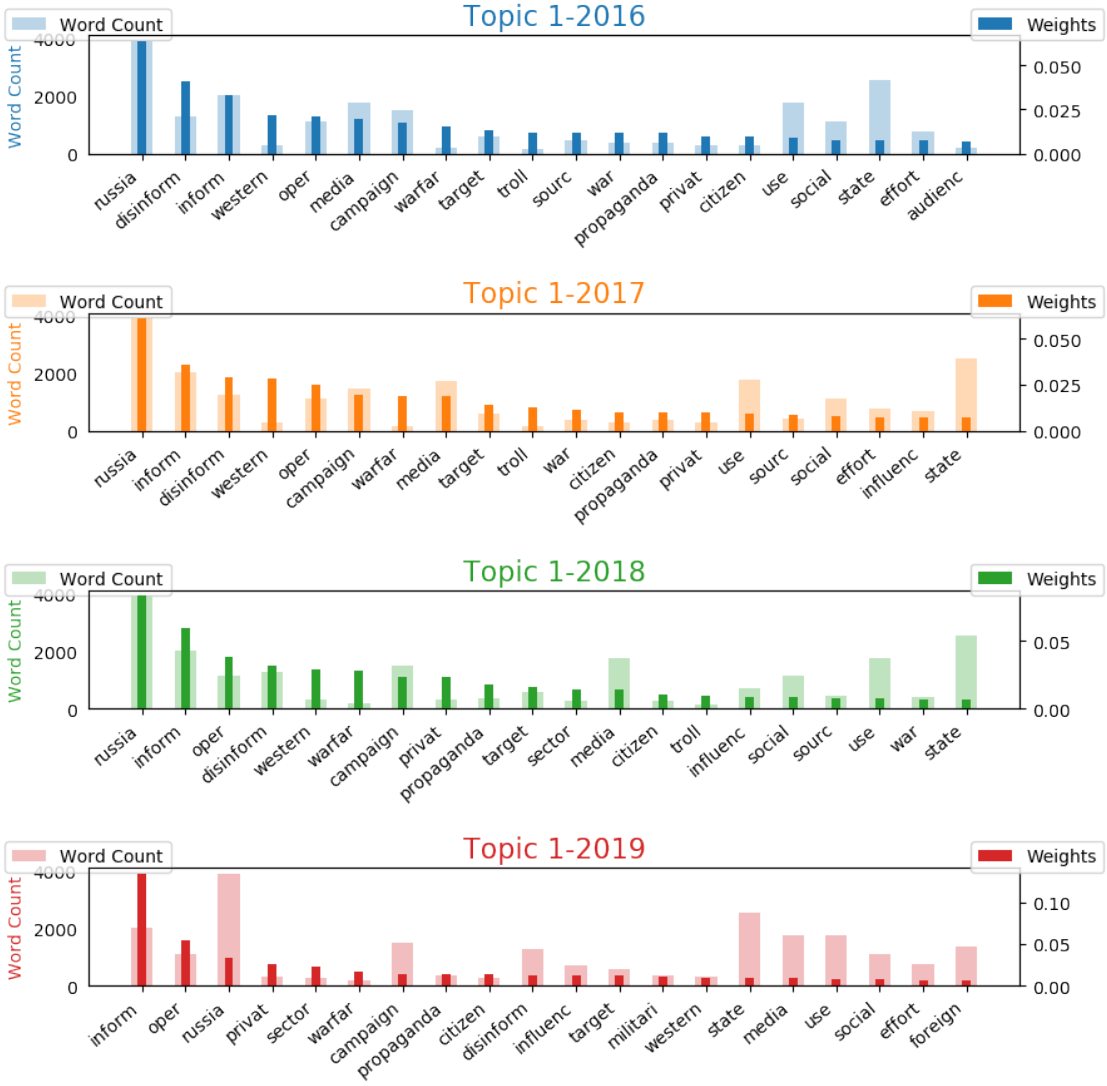
[147]:
```
# save model
pickle.dump(ldaseq_10, open("ldaseq_model_10_new.sav", 'wb'))
```

[148]:
```
# calculate coherence matrix
topics_dtm_101 = ldaseq_10.dtm_coherence(time=0)
topics_dtm_102 = ldaseq_10.dtm_coherence(time=1)
topics_dtm_103 = ldaseq_10.dtm_coherence(time=2)
topics_dtm_104 = ldaseq_10.dtm_coherence(time=3)
cm_DTM_101 = CoherenceModel(topics=topics_dtm_101, corpus=corpus_all,␣
 ↪dictionary=dic_all, coherence='u_mass')
cm_DTM_102 = CoherenceModel(topics=topics_dtm_102, corpus=corpus_all,␣
 ↪dictionary=dic_all, coherence='u_mass')
cm_DTM_103 = CoherenceModel(topics=topics_dtm_103, corpus=corpus_all,␣
 ↪dictionary=dic_all, coherence='u_mass')
cm_DTM_104 = CoherenceModel(topics=topics_dtm_104, corpus=corpus_all,␣
 ↪dictionary=dic_all, coherence='u_mass')
print ("U_mass topic coherence")
print ("DTM Python coherence is", cm_DTM_101.get_coherence(), ";", cm_DTM_102.
 ↪get_coherence(), ";",
      cm_DTM_103.get_coherence(), ";", cm_DTM_104.get_coherence())
```

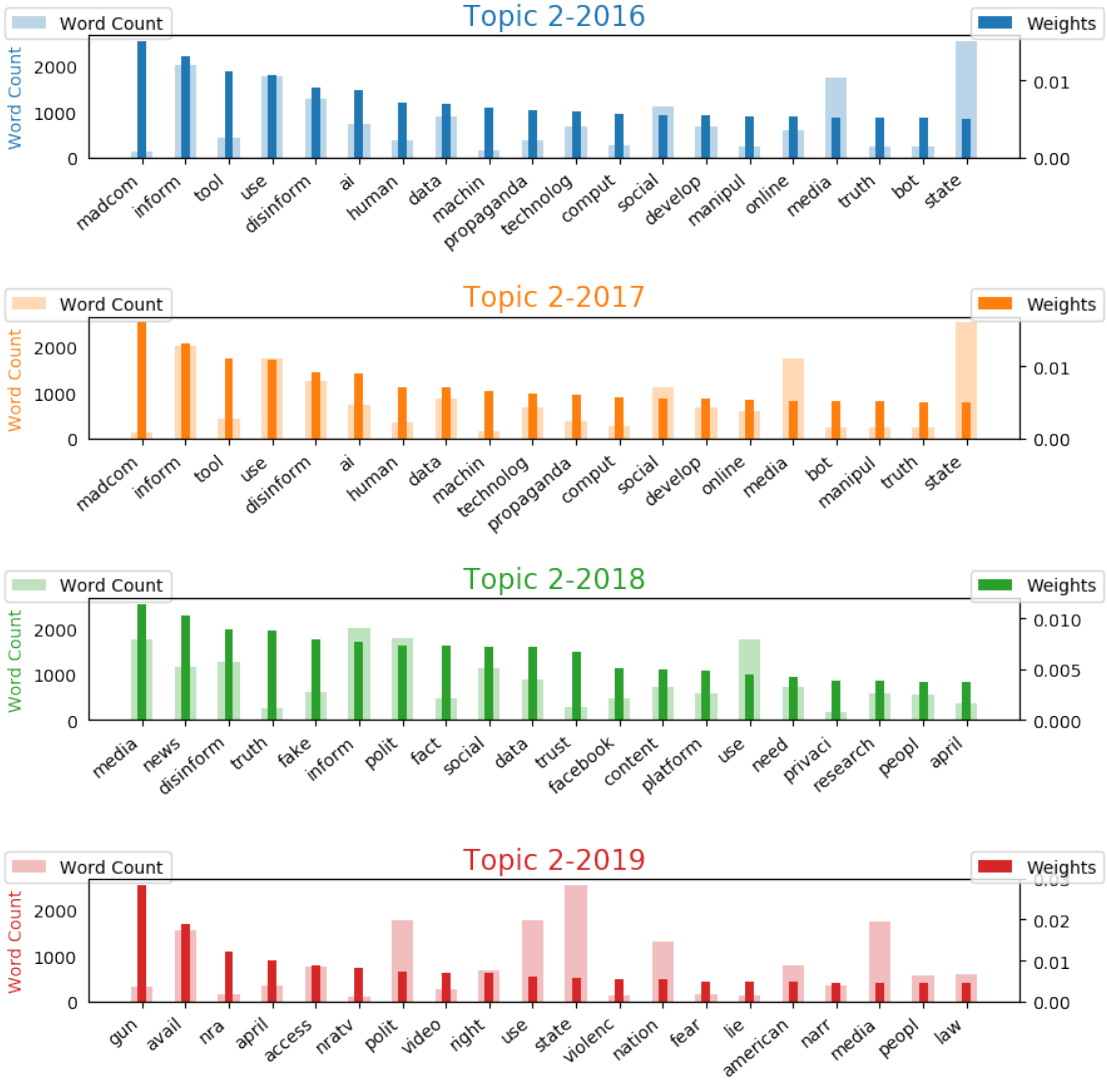
```
U_mass topic coherence
DTM Python coherence is -0.4123640368510368 ; -0.2848357912960775 ;
-0.2845824987809631 ; -0.33056695852714446
```

[149]:
```
# print topic evolution
first_topic_10 = ldaseq_10.print_topic_times(topic=0)
second_topic_10 = ldaseq_10.print_topic_times(topic=1)
third_topic_10 = ldaseq_10.print_topic_times(topic=2)
fourth_topic_10 = ldaseq_10.print_topic_times(topic=3)
fifth_topic_10 = ldaseq_10.print_topic_times(topic=4)
sixth_topic_10 = ldaseq_10.print_topic_times(topic=5)
seventh_topic_10 = ldaseq_10.print_topic_times(topic=6)
eighth_topic_10 = ldaseq_10.print_topic_times(topic=7)
ninth_topic_10 = ldaseq_10.print_topic_times(topic=8)
tenth_topic_10 = ldaseq_10.print_topic_times(topic=9)
```

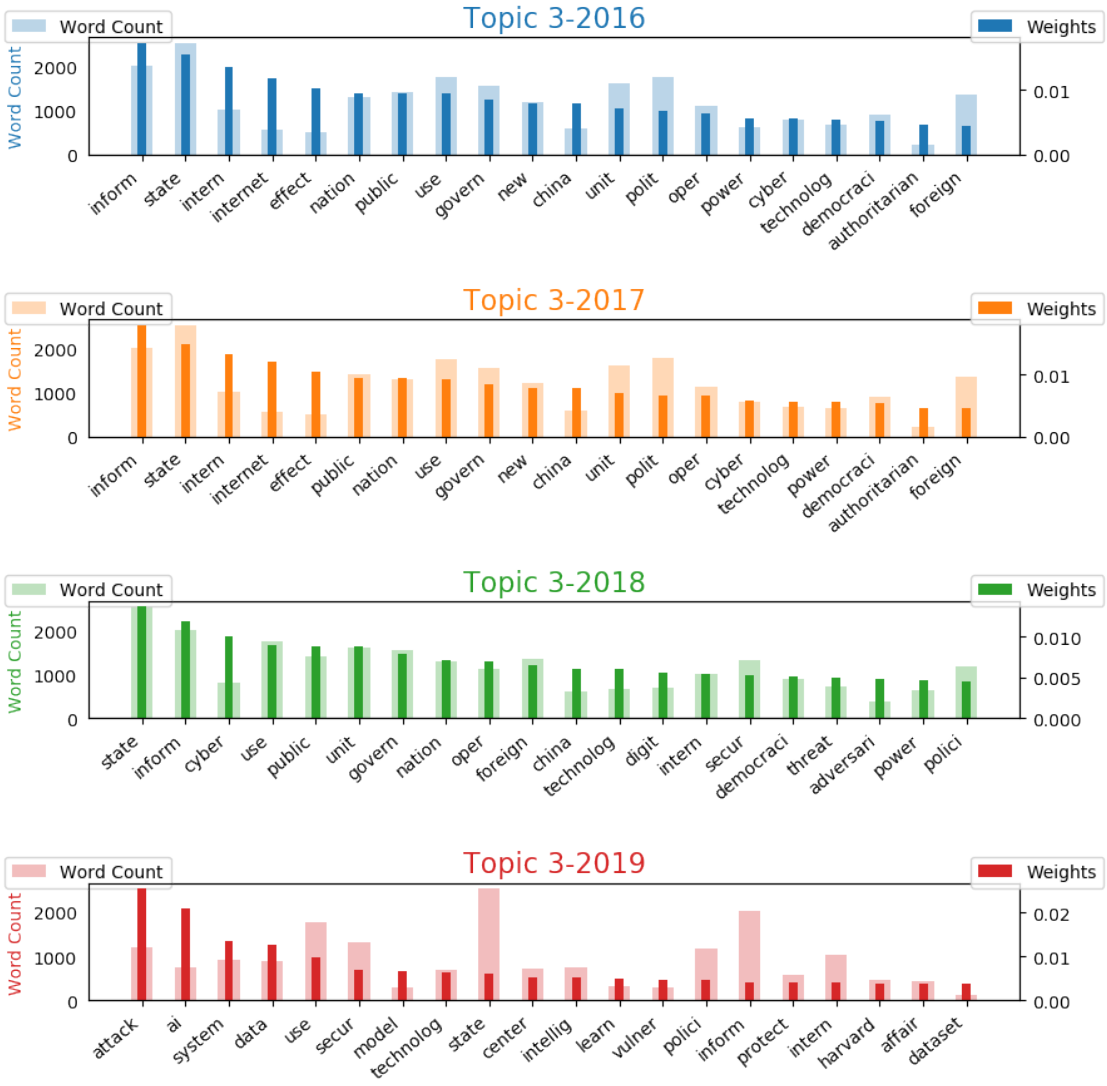
[168]:
```
#the first topic
vis_df_101 = construct_vis_df(processed_df, first_topic_10)
times_101 = ["Topic 1-2016", "Topic 1-2017",
         "Topic 1-2018", "Topic 1-2019"]
graph_topic_keyword_count(vis_df_101, 'Topic1', times_101)
```



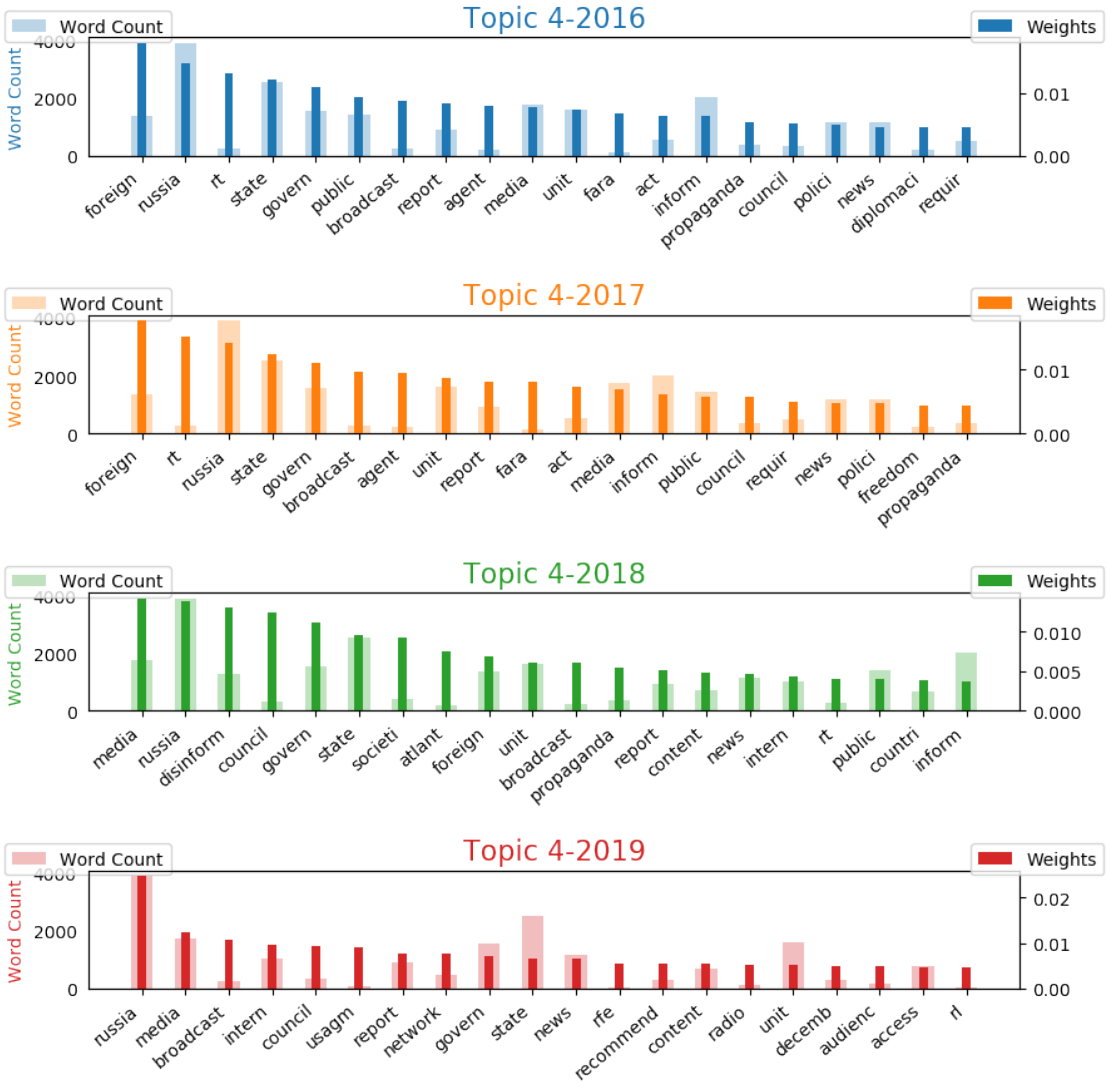
[150]:
```
#the second topic
vis_df_102 = construct_vis_df(processed_df, second_topic_10)
times_102 = ["Topic 2-2016", "Topic 2-2017",
         "Topic 2-2018", "Topic 2-2019"]
graph_topic_keyword_count(vis_df_102, 'Topic2', times_102)
```



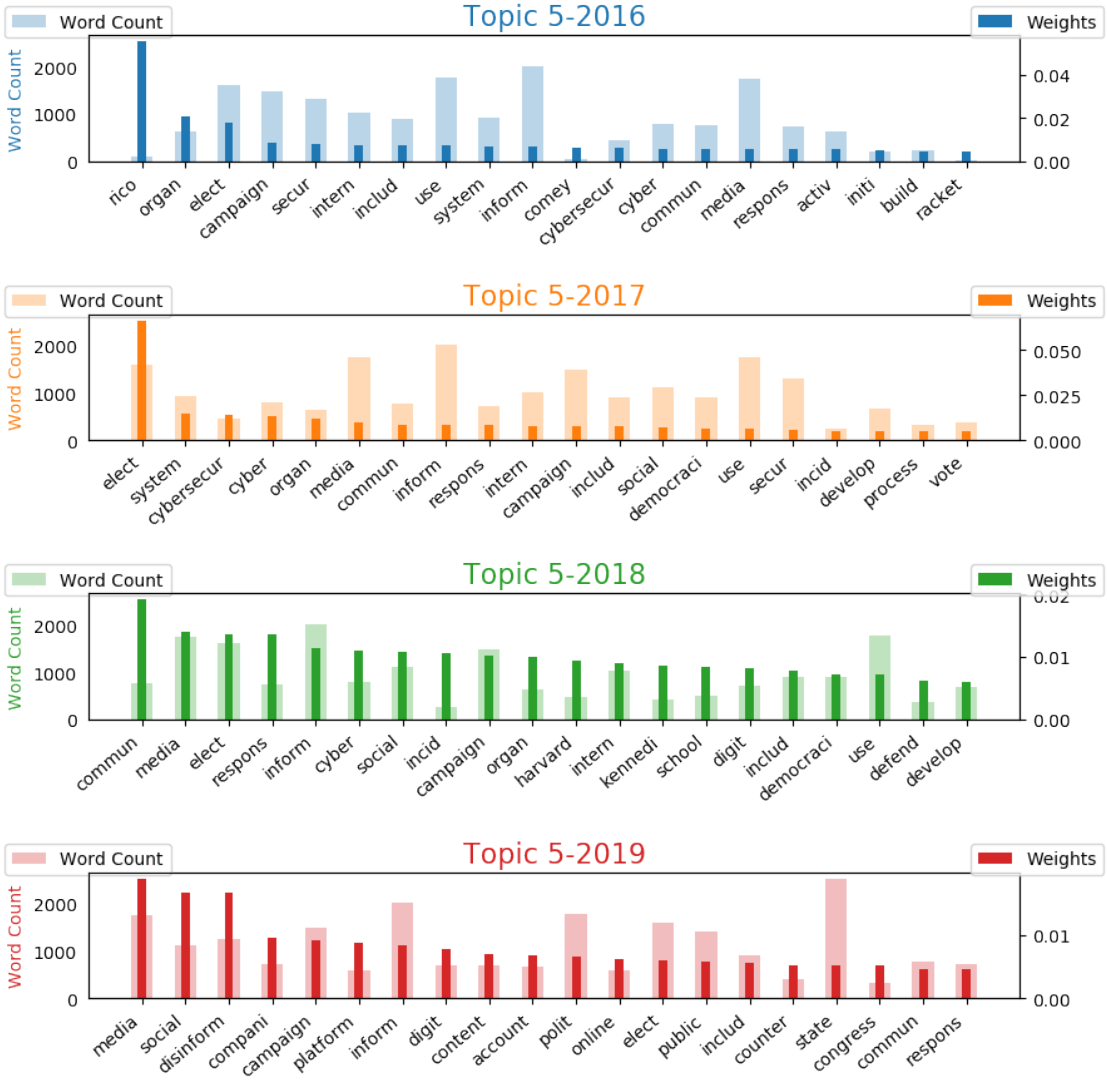
[151]:
```
#the third topic
vis_df_103 = construct_vis_df(processed_df, third_topic_10)
times_103 = ["Topic 3-2016", "Topic 3-2017",
         "Topic 3-2018", "Topic 3-2019"]
graph_topic_keyword_count(vis_df_103, 'Topic3', times_103)
```



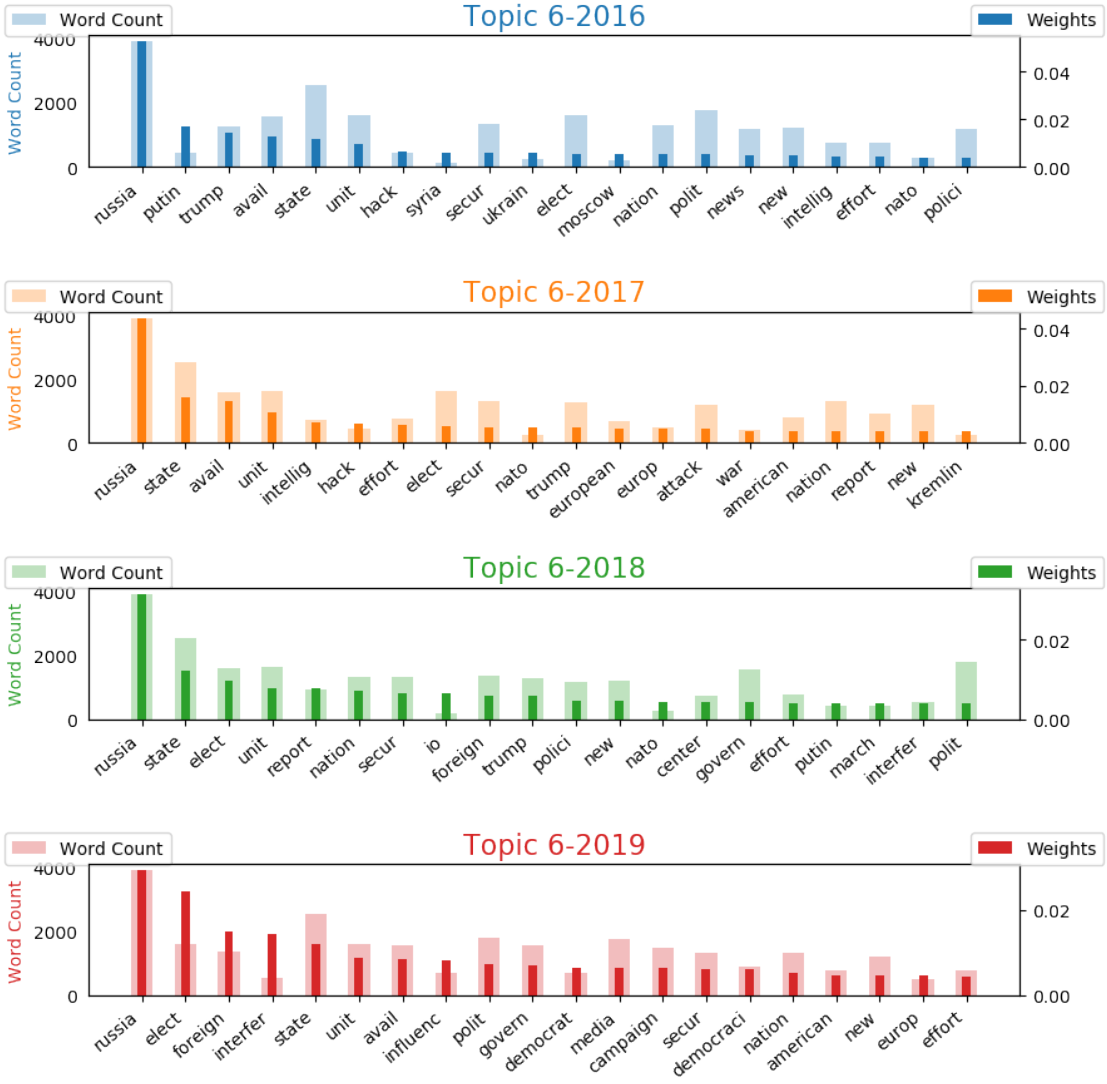
[152]:
```
#the fourth topic
vis_df_104 = construct_vis_df(processed_df, fourth_topic_10)
times_104 = ["Topic 4-2016", "Topic 4-2017",
         "Topic 4-2018", "Topic 4-2019"]
graph_topic_keyword_count(vis_df_104, 'Topic4', times_104)
```



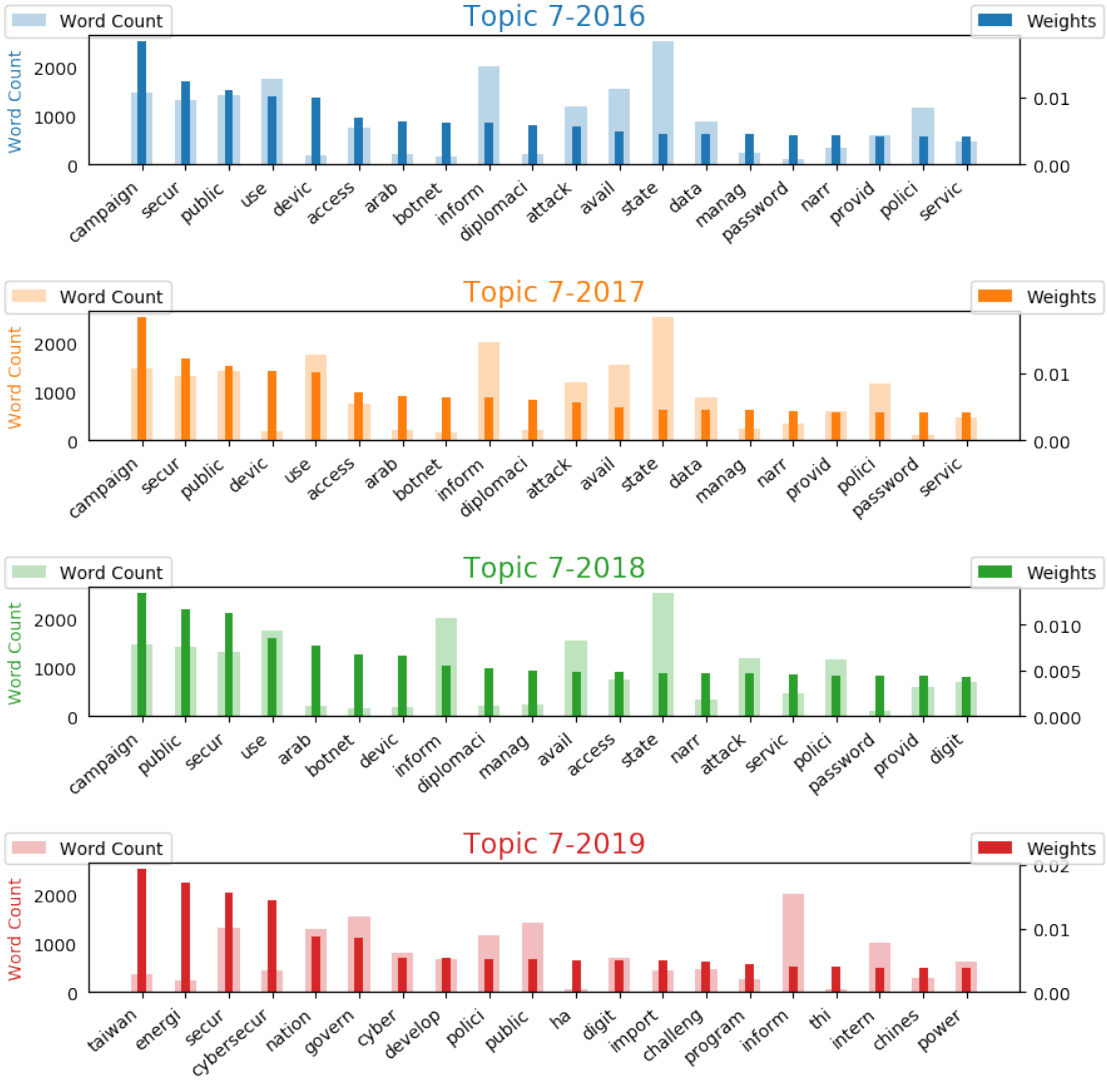
[153]:
```
#the fifth topic
vis_df_105 = construct_vis_df(processed_df, fifth_topic_10)
times_105 = ["Topic 5-2016", "Topic 5-2017",
         "Topic 5-2018", "Topic 5-2019"]
graph_topic_keyword_count(vis_df_105, 'Topic5', times_105)
```



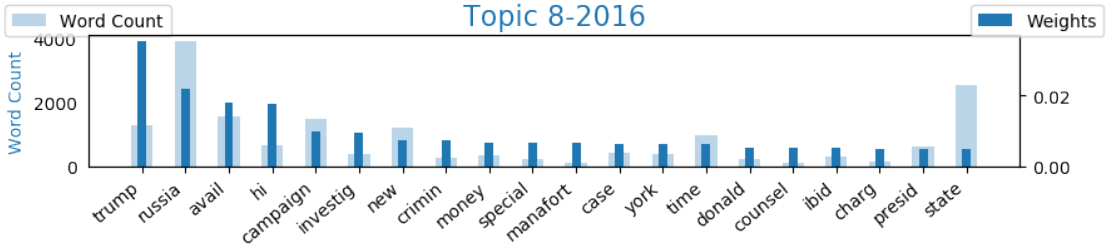
[154]:
```
#the sixth topic
vis_df_106 = construct_vis_df(processed_df, sixth_topic_10)
times_106 = ["Topic 6-2016", "Topic 6-2017",
         "Topic 6-2018", "Topic 6-2019"]
graph_topic_keyword_count(vis_df_106, 'Topic6', times_106)
```

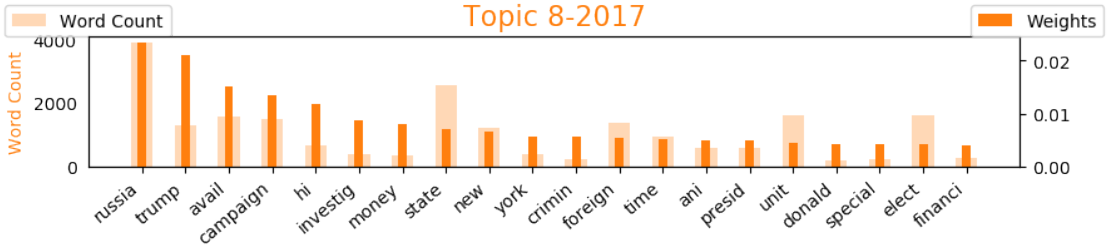


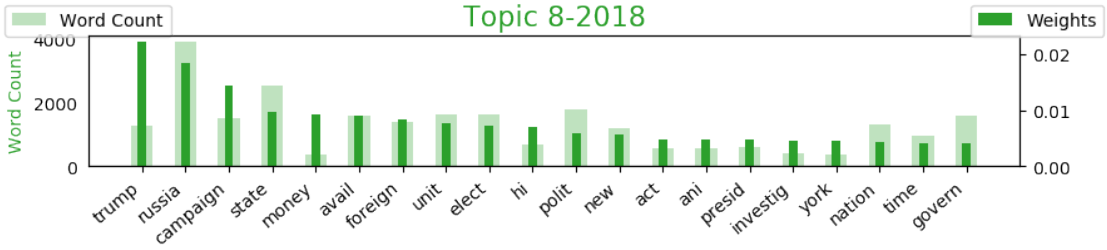
[155]:
```
#the seventh topic
vis_df_107 = construct_vis_df(processed_df, seventh_topic_10)
times_107 = ["Topic 7-2016", "Topic 7-2017",
         "Topic 7-2018", "Topic 7-2019"]
graph_topic_keyword_count(vis_df_107, 'Topic7', times_107)
```

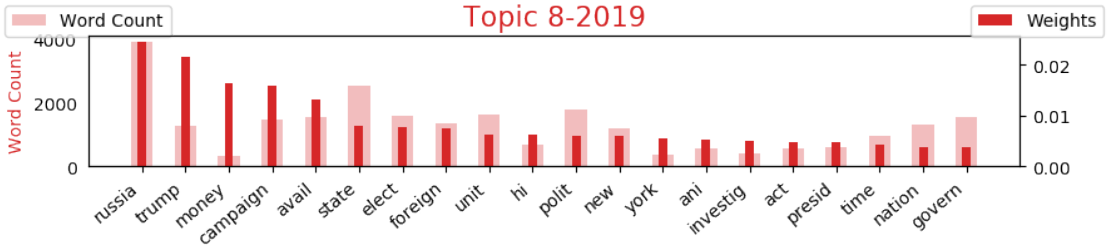


[156]:
```
#the eighth topic
vis_df_108 = construct_vis_df(processed_df, eighth_topic_10)
times_108 = ["Topic 8-2016", "Topic 8-2017",
         "Topic 8-2018", "Topic 8-2019"]
graph_topic_keyword_count(vis_df_108, 'Topic8', times_108)
```

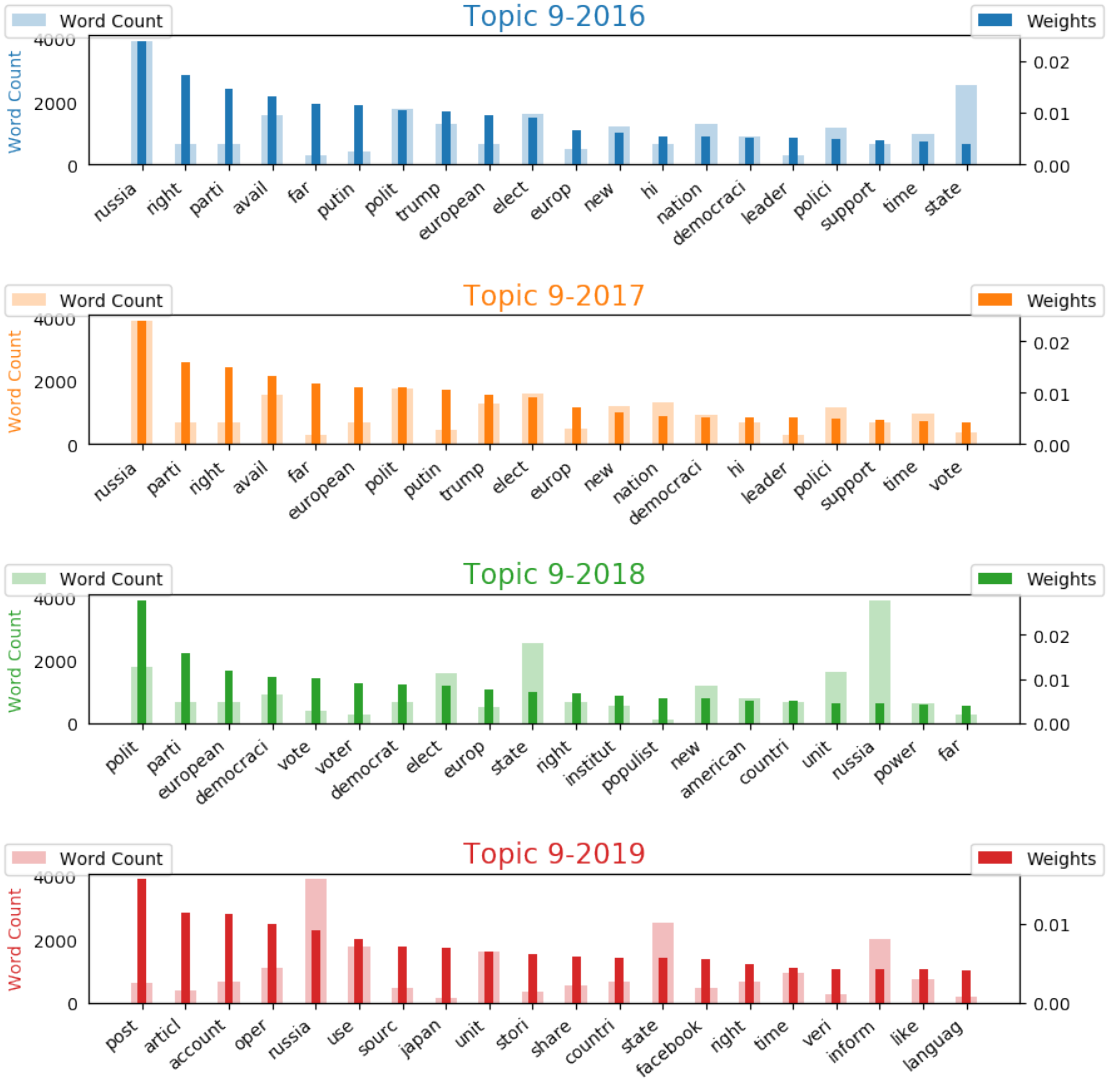




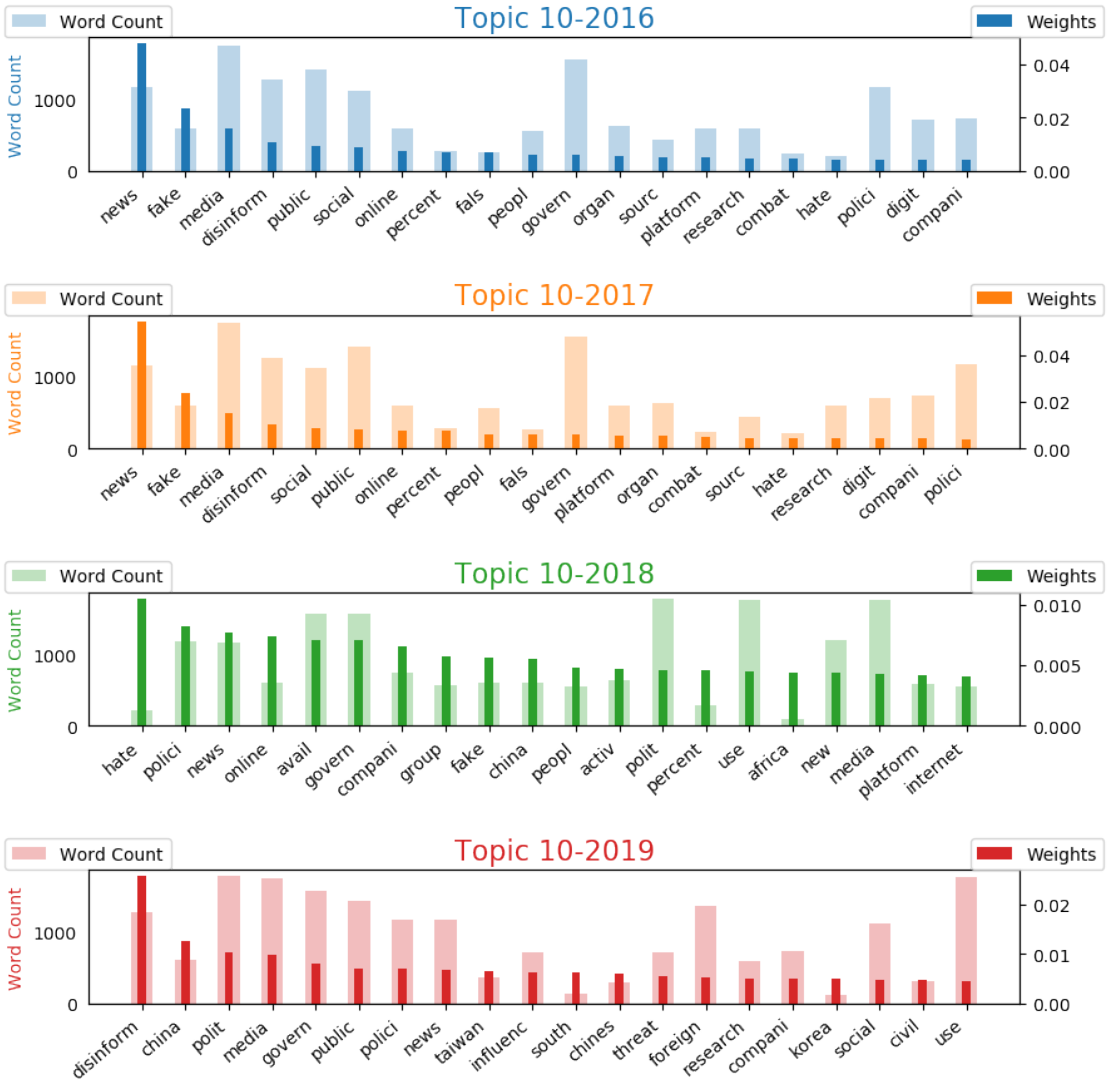




[157]:
```
#the ninth topic
vis_df_109 = construct_vis_df(processed_df, ninth_topic_10)
times_109 = ["Topic 9-2016", "Topic 9-2017",
         "Topic 9-2018", "Topic 9-2019"]
graph_topic_keyword_count(vis_df_109, 'Topic8', times_109)
```



[158]:
```
#the tenth topic
vis_df_1010 = construct_vis_df(processed_df, tenth_topic_10)
times_1010 = ["Topic 10-2016", "Topic 10-2017",
         "Topic 10-2018", "Topic 10-2019"]
graph_topic_keyword_count(vis_df_1010, 'Topic8', times_1010)
```



0.1.4 4. Compare topic coherence score for six models

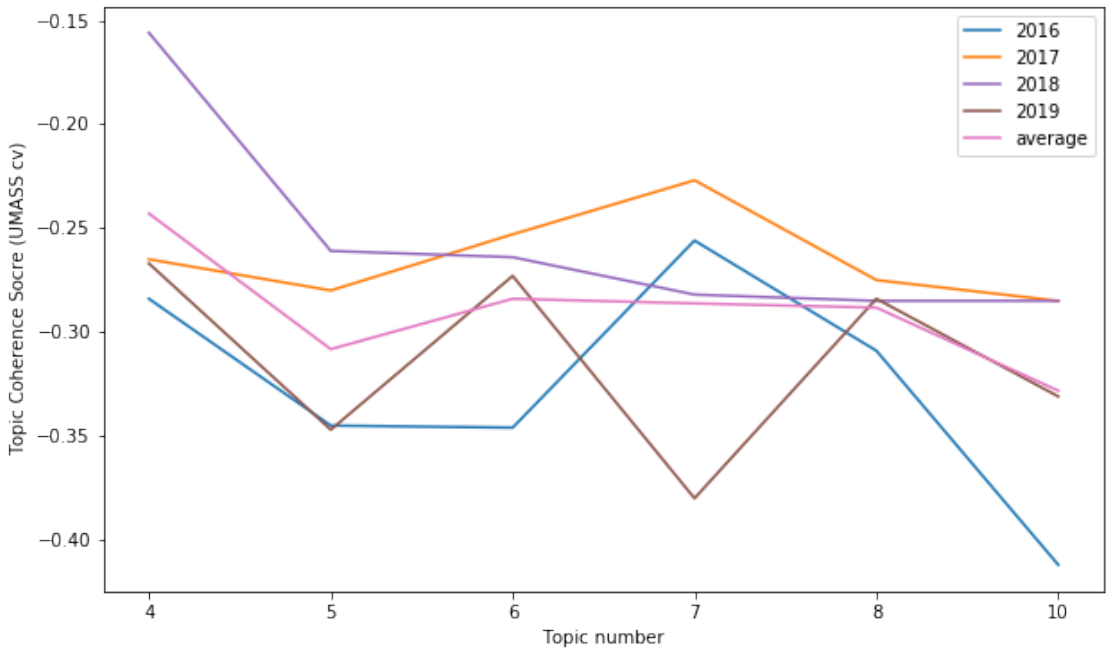
[179]:
```
time_0 = [-0.284, -0.345, -0.346, -0.256, -0.309, -0.412]
time_1 = [-0.265,-0.280,   -0.253, -0.227, -0.275, -0.285]
time_2 = [-0.156, -0.261,  -0.264, -0.282, -0.285, -0.285]
time_3 = [-0.267, -0.347, -0.273, -0.380, -0.284, -0.331]
average_time = []
for i in range(6):
    sum_ch = time_0[i] + time_1[i] + time_2[i] + time_3[i]
    average_time.append(sum_ch/4)
all_tine = [time_0, time_1, time_2, time_3, average_time]
umass_ch = pd.DataFrame(all_tine, columns=['4', '5', '6', '7', '8', '10'])
```

[180]:
```
umass_ch_trans = umass_ch.T.reset_index()
umass_ch_trans
```

[180]:

	index	0	1	2	3	4
0	4	-0.284	-0.265	-0.156	-0.267	-0.24300
1	5	-0.345	-0.280	-0.261	-0.347	-0.30825
2	6	-0.346	-0.253	-0.264	-0.273	-0.28400
3	7	-0.256	-0.227	-0.282	-0.380	-0.28625
4	8	-0.309	-0.275	-0.285	-0.284	-0.28825
5	10	-0.412	-0.285	-0.285	-0.331	-0.32825

[181]:
```
plt.figure(figsize=(10, 6))
plt.plot(umass_ch_trans['index'], umass_ch_trans[0], color='C0', label='2016')
plt.plot(umass_ch_trans['index'], umass_ch_trans[1], color='C1', label='2017')
plt.plot(umass_ch_trans['index'], umass_ch_trans[2], color='C4', label='2018')
plt.plot(umass_ch_trans['index'], umass_ch_trans[3], color='C5', label='2019')
plt.plot(umass_ch_trans['index'], umass_ch_trans[4], color='C6',␣
 ↪label='average')
plt.xlabel("Topic number")
plt.ylabel("Topic Coherence Socre (UMASS cv)")
plt.legend(loc='best')
plt.savefig('./visualization_2/TOPIC_COHERENCE')
plt.show()
```



0.1.5 5. Validate the 4-topics model with a subset of data

[21]:
```
# read in csv file for the subset of data
data = pd.read_csv('./valid_set.csv')
# sort by year
sorted_data = data.sort_values(by=['year'])
# change index
sorted_data = sorted_data.reset_index()
sorted_data = sorted_data.drop(columns=['index', 'Unnamed: 0'])
```

[22]:
```
# preprocess the data
processed_df = sorted_data['text'].map(preprocess)
```

[23]:
```
# prepare Document-Term Matrix for the DTA model
# Create Dictionaries for unique word counts of each decade
dic_all = corpora.Dictionary(processed_df)

# Create Corpus: Term Document Frequency
corpus_all = [dic_all.doc2bow(text) for text in processed_df]
```

[14]:
```
# fit the 4-topic model
start = time.time()
ldaseq_4 = ldaseqmodel.LdaSeqModel(corpus=corpus_all, id2word=dic_all,␣
 ↪time_slice=time_slice,
                                   num_topics=4, chain_variance=0.1)
end = time.time()
print(end - start)
```

```
/Users/ditong/anaconda3/lib/python3.7/site-
packages/gensim/models/ldaseqmodel.py:230: RuntimeWarning: divide by zero
encountered in double_scalars
  convergence = np.fabs((bound - old_bound) / old_bound)

1665.3317959308624
```

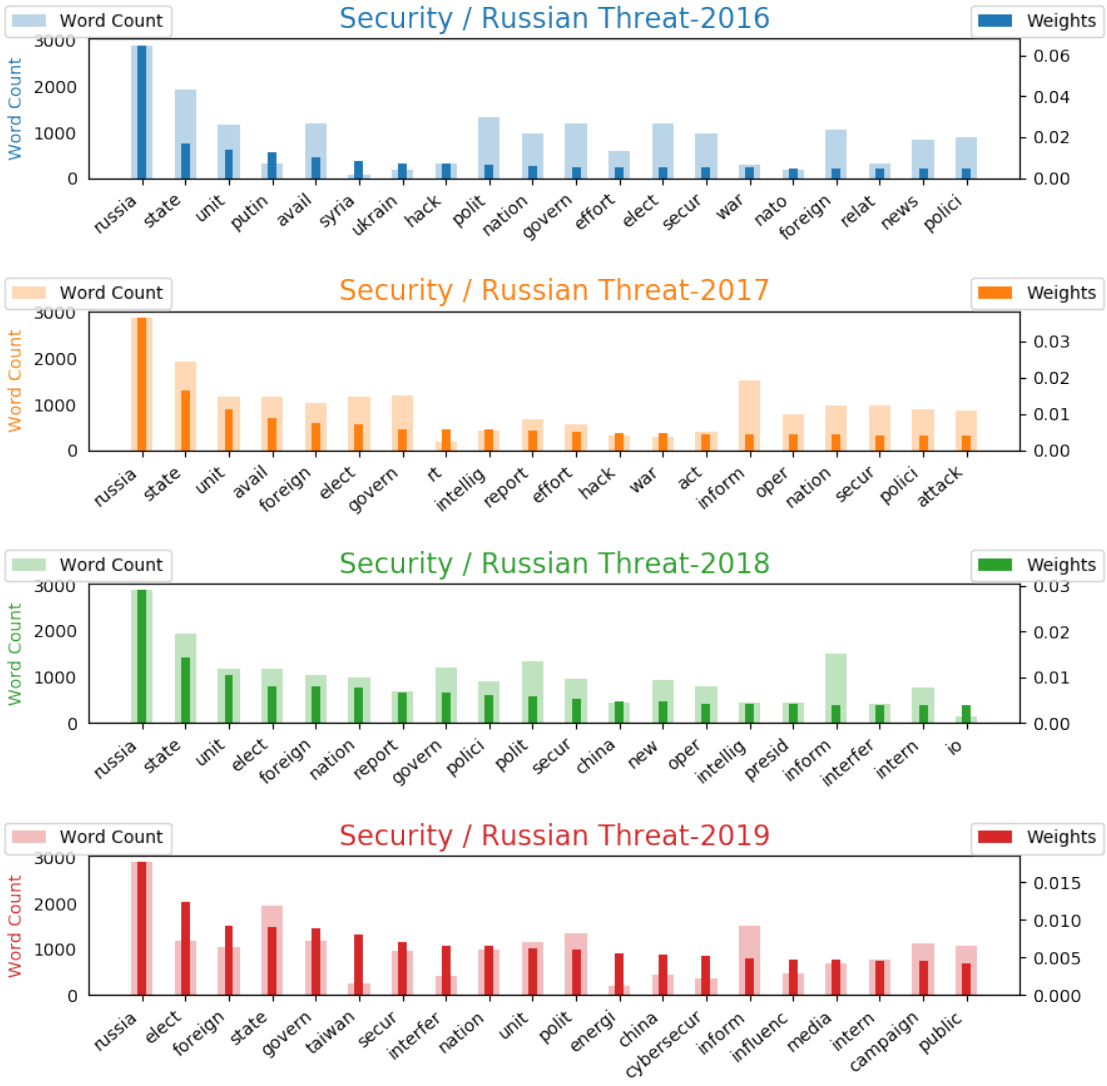
[15]:
```
# save model
pickle.dump(ldaseq_4, open("ldaseq_model_4_validate.sav", 'wb'))
```

[24]:
```
# load model
ldaseq_4 = pickle.load(open("ldaseq_model_4_validate.sav", 'rb'))
```

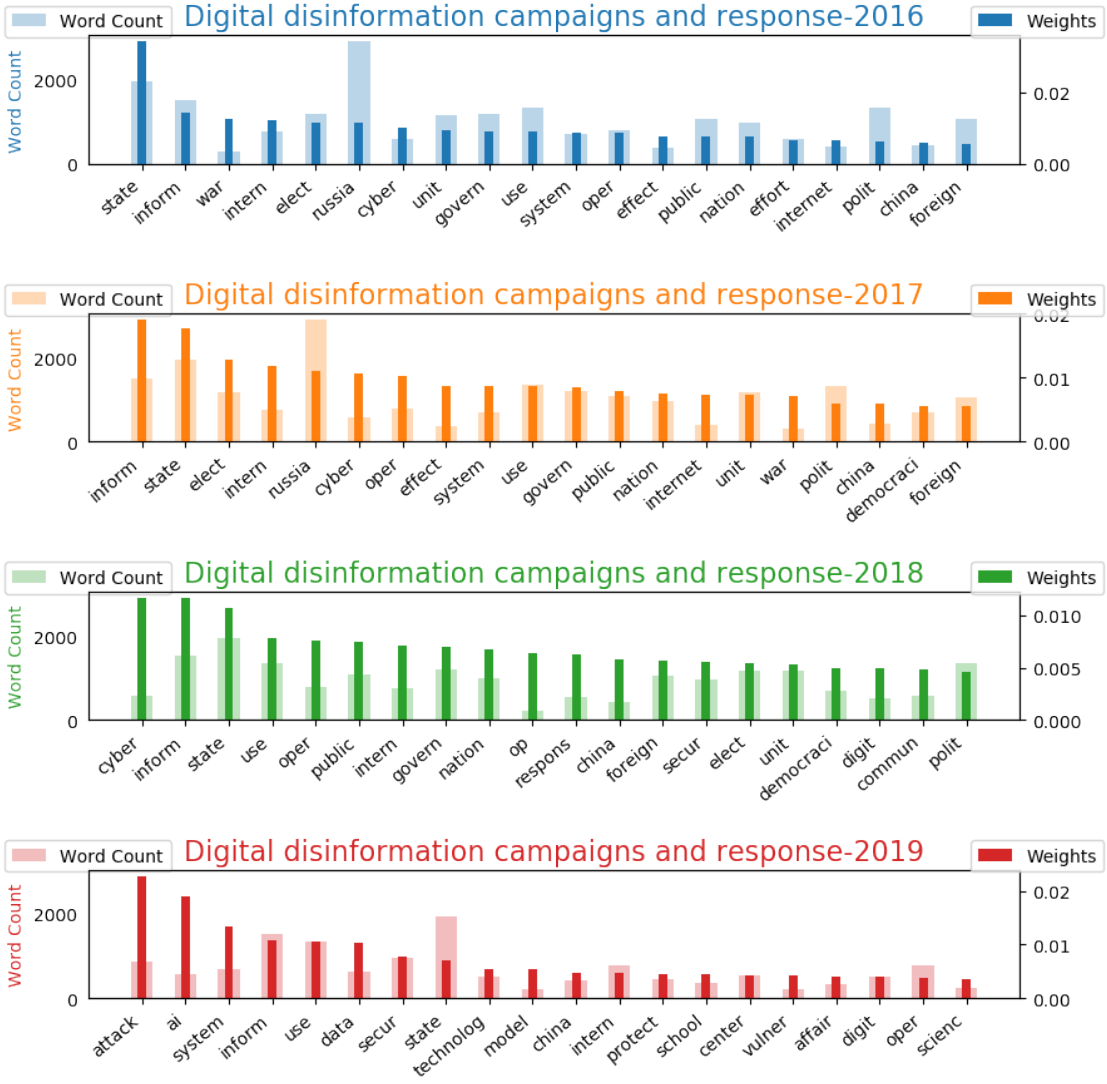
[25]:
```
# print topic evolution
first_topic_4 = ldaseq_4.print_topic_times(topic=0)
second_topic_4 = ldaseq_4.print_topic_times(topic=1)
third_topic_4 = ldaseq_4.print_topic_times(topic=2)
fourth_topic_4 = ldaseq_4.print_topic_times(topic=3)
```

[17]:
```
#the first topic
vis_df_41 = construct_vis_df(processed_df, first_topic_4)
times_41 = ["Security / Russian Threat-2016", "Security / Russian Threat-2017",
```

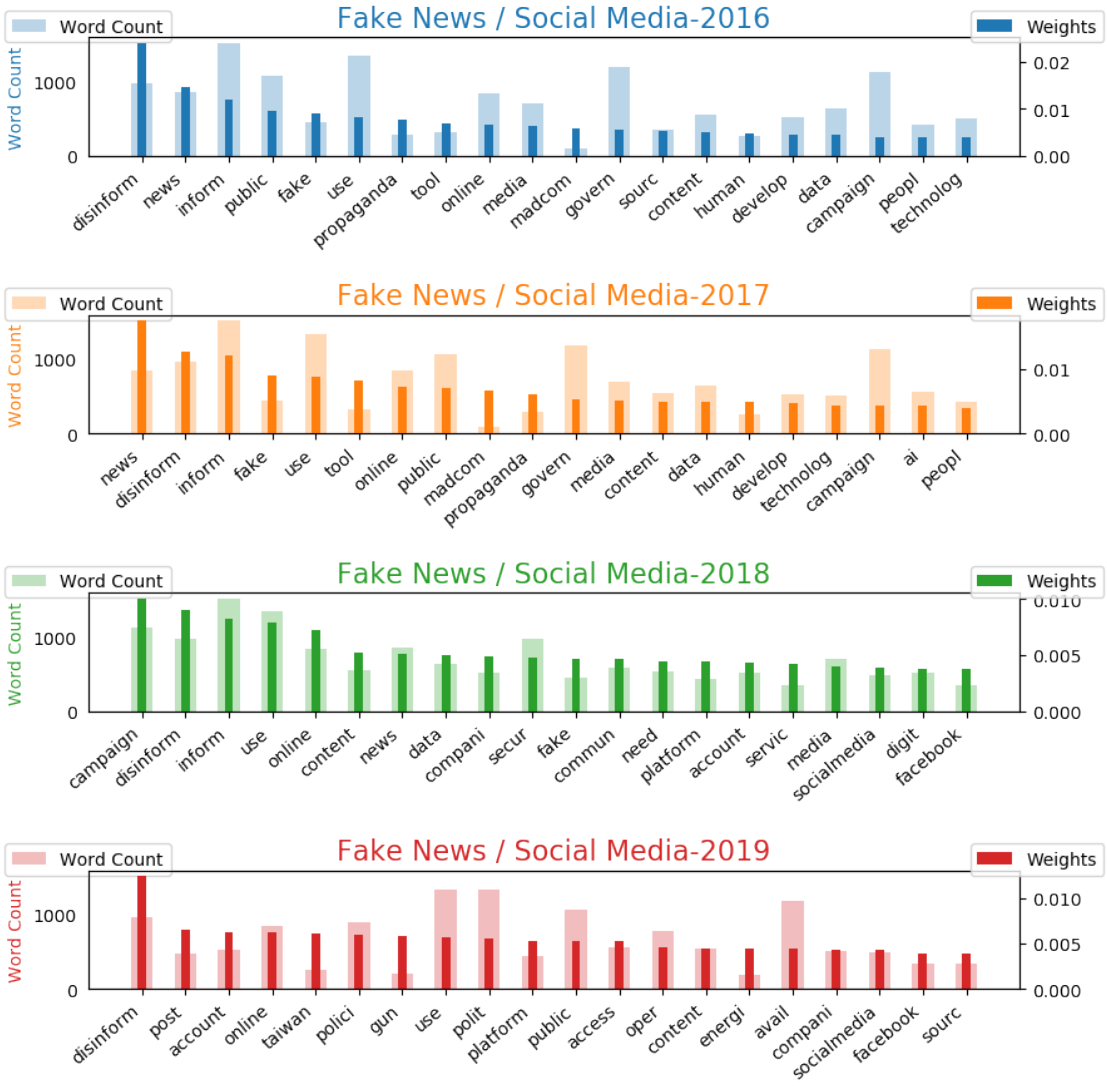
```
        "Security / Russian Threat-2018", "Security / Russian Threat-2019"]
graph_topic_keyword_count(vis_df_41, 'Topic1', times_41)
```



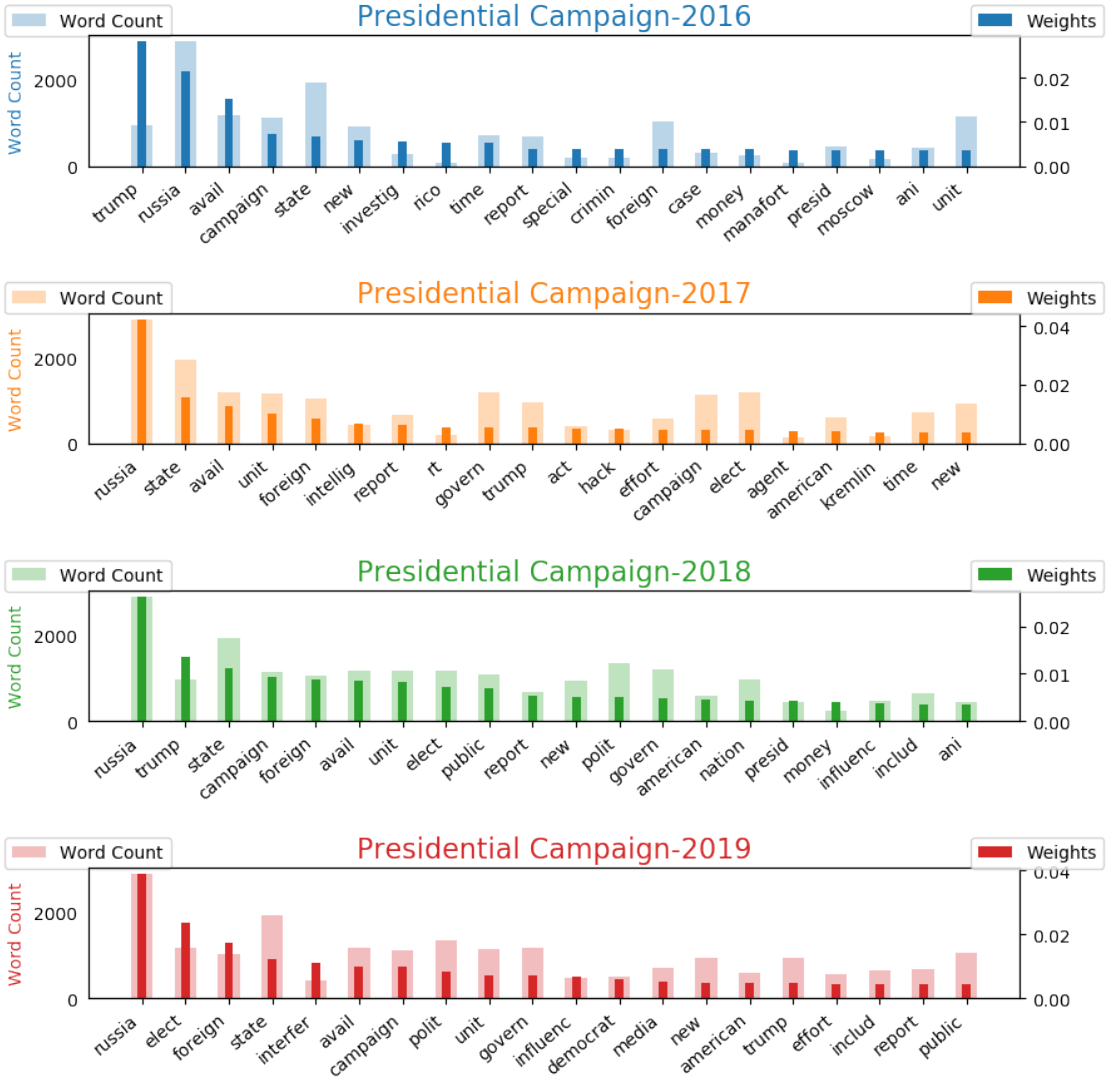
[26]:
```
#the second topic
vis_df_42 = construct_vis_df(processed_df, second_topic_4)
times_42 = ["Digital disinformation campaigns and response-2016", "Digital␣
 ↪disinformation campaigns and response-2017",
        "Digital disinformation campaigns and response-2018", "Digital␣
 ↪disinformation campaigns and response-2019"]
graph_topic_keyword_count(vis_df_42, 'Topic2', times_42)
```



[27]:
```
#the third topic
vis_df_43 = construct_vis_df(processed_df, third_topic_4)
times_43 = ["Fake News / Social Media-2016", "Fake News / Social Media-2017",
        "Fake News / Social Media-2018", "Fake News / Social Media-2019"]
graph_topic_keyword_count(vis_df_43, 'Topic3', times_43)
```



[28]:
```
#the fourth topic
vis_df_44 = construct_vis_df(processed_df, fourth_topic_4)
times_44 = ["Presidential Campaign-2016", "Presidential Campaign-2017",
         "Presidential Campaign-2018", "Presidential Campaign-2019"]
graph_topic_keyword_count(vis_df_44, 'Topic4', times_44)
```



[26]:
```
doc_topic_4 = []
for doc_num in range(len(sorted_data)):
    doc = ldaseq_4.doc_topics(doc_num)
    doc_topic_4.append(doc)
topic_df_4 = pd.DataFrame(doc_topic_4, columns=['t1', 't2', 't3', 't4'])
topic_com_df_4 = pd.concat([sorted_data, topic_df_4], axis=1, sort=False)
grouped_df_4 = topic_com_df_4.groupby(['year']).sum().reset_index()
grouped_df_av_4 = topic_com_df_4.groupby(['year']).mean().reset_index()
```

[27]:
```
topics = ['t1', 't3', 't4', 't2']
labels = ['Security / Russian Threat', 'Fake News / Social Media',␣
 ↪'Presidential Campaign',
          'Digital disinformation campaigns and response']
ylabel = "Average Topic Proportion"
```

```
topic_prevalence_evolution(grouped_df_av_4, topics, labels, ylabel)
```