1. **Code:**Please include your code in the form of a Github repo link in a footnote somewhere in your synopsis. If you make changes to the code after the January 3rd synopsis deadline, you should push these to Github.
2. **Synopsis content:**The synopsis should - at time of submission - include implementation results. I.e. it cannot be simply a study plan/design, although you may talk about "next steps". If you make further progress after January 3rd and come prepared to discuss these results in the oral presentation, that's great (it will likely help you anticipate some of our questions, which may have a positive impact on your grade). But the synopsis should still stand up on its own.

The final project is to either build a natural language processing system, or apply one for some task. The project must use or develop a dataset, and report empirical results or analyses with the dataset. It may use machine learning or rule-based approaches. It may use any type of open-source or widely available software.You can choose to emphasize:

* Implementing and developing algorithms and features.
* Defining a new linguistic / text analysis task, and tackling it with off-the-shelf NLP software.
* Collect and explore a new textual dataset to address research hypotheses about it.

Different projects will have different balances of these three things.

The key requirement is to investigate, analyze, and come to research findings about new methods, or insights about previously existing methods.

The synopsis can be done as an individual or group assignment. If it is done as a group assignment, the final product must  
(i) form a coherent text and (ii) be organized so that it is possible to make individual assessments of the students contributing.  
In other words, the contribution of each individual student to the whole assignment must be clearly delineated (excluding the introduction, conclusion and bibliography).  
A maximum of three students can take part in a group assignment.

* Length of individual synopsis: 4-7 standard pages (not including code and figures)
* Length of synopsis for 2 students: 8-14 standard pages (not including code and figures)
* Length of synopsis for 3 students: 12-21 standard pages (not including code and figures)

The final synopsis  describes your project and final results. Conceptually, it should include the content of both the proposal and progress report, though they will be changed. The final report describes and motivates the problem, places it in context of related work, describes the dataset and your approach, and reports results with discussion and thoughts for future work.

Here is a sample outline for your final report. There are different possible ways to structure it (for example, if you can, you can weave related work into the other sections), but we suggest you follow this outline unless you have substantial prior experience writing technical reports and research papers.

* **Abstract:** summarize the main components of your work in one paragraph (no more than 5 sentences). What problem are you solving? What is the key to your approach? What results did you achieve? Your abstract should draw the reader in and interest them in reading the rest of your paper to understand the details of your work.
* **Introduction:** explain the problem, motivate it (why is it important?), and briefly describe your approach. State a research question that your project seeks to answer: what are you trying to learn from this research project? You may also report some of your results without discussing the details of your method.
* **Data:** Describe the dataset that you are using.
* **Method:** Describe your approach to handling the problem. This should should include any models you used and any modeling assumptions you made. If you’ve developed new models for this project, you may even want to split a description/analysis of your models into its own section.
* **Results:** Describe the experiments you ran and identify your baseline method(s). Include the results you achieved with the various methods you are comparing making. This section will probably also include some figures that succinctly summarize your results. Analyze your results (including your models). If you did exploratory analysis or a significant amount of feature engineering, your analysis may merit its own section. After reading this section (and your dataset and methods), an interested reader should be able to duplicate your experiments and results.
* **Discussion and Future Work:** discuss any implications of your analysis for the problem as a whole, and what are the next steps for future work. Any other concluding remarks should go here.

A Matter of Tech Security

Predicting Trends from Podcasts

# Abstract

The podcast Security Now has been running every week since mid-2005. Here it is used as a novel source of NLP data. Using LDA, transcripts of the podcasts are used to investigate how topics on tech security change over time. We find that

# Introduction

Ever since the widespread introduction of the personal computer in the early 1980’s the share of data we put online has and still continues to grow exponentially every day (Schultz, 2019; ‘Timeline of Computer History’, n.d.). This brings up the issue of how to best safeguard our data and what rules to follow as we “travel online”. And while some still think their birthday is a perfectly fine password for anything from social media accounts to web banking, many are beginning to see the necessity for properly protecting their data. This tendency became even more evident with the instantiation of the General Data Protection Regulation (GDPR) in May of 2018. The GDPR helped introduce the subject of internet security to the casual user and caused awareness of personal data protection to rise substantially all across Europe (Burgess, 2019). But for some, internet security is not a new phenomenon at all.

## Case Study: Security Now Podcast

The podcast ‘Security Now’ has been running continuously since mid-2005. Its two hosts, Steve Gibson and Leo Laporte, in their own words “*spend somewhat shy of two hours each week to discuss important issues of personal computer security.”* (‘GRC | Security Now! Episode Archive’, n.d.).

Here we use transcripts of the podcasts to investigate whether changes in trends are observable over time in the wide field of personal computing. In addition, we are interested in finding out if any of the trends or “topics” can be said to have been predicted preemptively by these two experts in the field.

## Quantitative Text Analysis with Topic Modelling

To find out which topics have been prevalent in the area of tech security over time, we use LDA topic modelling, a quantitative text analysis method first described in 2003 by Blei, Ng and Jordan. LDA stands for *Latent Dirichlet Allocation* and is a generative probabilistic model.This means that the model views texts as collections of words sampled from a probability distribution. Each document in a corpus is seen as a probability distribution over a fixed set of topics. These topics are themselves probability distributions over all the words in the corpus (Blei, Ng, & Jordan, 2003).

The term ‘latent’ refers to the fact that topics are assumed to be underlyingly present in the corpus at all times but only become accessible as the topic model is performed. As the model is unsupervised these topics are automatically created based on the documents. ‘Dirichlet’[[1]](#footnote-1) is the name of any probability distribution consisting of a range of probability distributions. As previously mentioned, this is exactly the case in LDA where documents consist of topic distributions, which themselves consist of word distributions (Fig 1(Li, Wang, Yang, Rong, & Yang, 2017)) (Ganegedara, 2019).

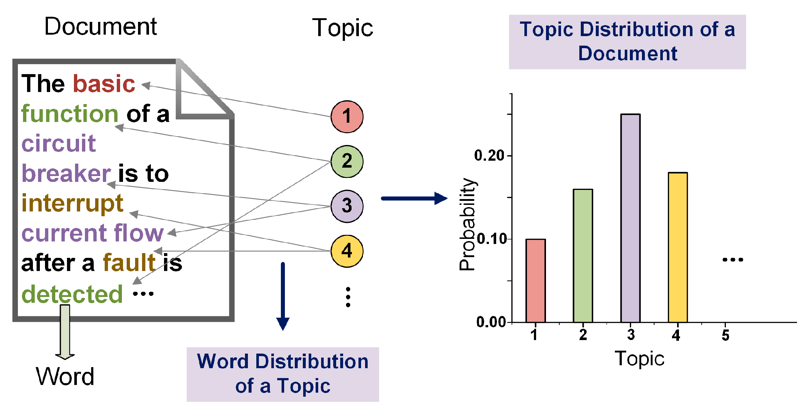


Figure 1. Illustrative example of Latent Dirichlet Allocation (LDA). Documents consist of different proportions of topics, which themselves are probability distributions of all words in the corpus. From (Li, Wang, Yang, Rong, & Yang, 2017)

## LDA in Research

LDA is not only used for text analysis but has been used to investigate a diverse range of research questions concerning bioinformatics, marketing intelligence, image classification and more. Text classification, however, is still by far the its most widely used application (Chong, Blei, & Li, 2009; Koks, 2019; Song & Kim, 2013).

In 2016 a group of researchers used LDA to investigate how coverage of nuclear technology in the *New York Times* had changed over time since 1945. They found clear trends of topic fluctuations over time and noted on LDA’s usefulness in providing the researcher with a quick overview over large collections of documents (Jacobi, Atteveldt, & Welbers, 2016).

The aim of any *quantitative* text analysis, as opposed to traditional *qualitative* text analysis, is to make *statistical* inferences from *large text corpora*. From this description, it is clear that such inferences are better suited for computers than humans. The first computerized text analysis program “The General Inquirer” was invented at Harvard in 1966 (Stone, Dunphy, & Smith, 1966). Around 1990 the term *Latent Semantic Indexing* (LSI) began making its way into the scientific literature on language and information sciences. The power of this new method lay in its ability to reduce “noise” in text retrieval by assuming an underlying (or “latent”) semantic word structure. ~~In other words search inquiries needed to be less specific, as documents were sought after in a context manner rather than by direct word-matching~~ (Deerwester, Dumais, Furnas, Landauer, & Harshman, 1990; Dumais, Letsche, Littman, & Kandauer, 1997).

Since then a wealth of computerized

Now, the combination of computers with massive computing power and the increasing digitalization of data has caused a change in the way text analysis is possible (Mehl, 2006).

Probability distributions rather than strict wod frequencies (<https://www.youtube.com/watch?v=DWJYZq_fQ2A>)

LSI (also known as Latent Semantic Analysis, LSA) learns latent topics by performing a matrix decomposition (SVD) on the term-document matrix.  
  
LDA is a generative probabilistic model, that assumes a Dirichlet prior over the latent topics.  
  
In practice, LSI is much faster to train than LDA, but has lower accuracy.

LDA states that each document in a corpus is a combination of a fixed number of topics. A topic has a probability of generating various words, where the words are all the observed words in the corpus.

# Materials and Methods

Below we present the data and methods used to arrive at our results. We begin by describing the dataset and the analysis pipeline. Following this, the method for model specification and model selection is presented and discussed. In the end, the chosen model is presented.

## Data statement

Data consisted of transcripts of 730 ‘Security Now’ podcast episodes, scraped from the public website www.grc.com/securitynow.htm. Data was gathered strictly for academic research purposes. Speaker demographic was predominantly homogenic (white, male, American, 30+) and consistent throughout the dataset. The texts were of semi-unstructured, conversational nature and the topic range was narrow; each episode discusses issues of personal computer security, but due to the format are sometimes more private in nature. Data was scraped for the period August 2015 - October 2019.

## GitHub Repository

All scripts used for analysis can be found in the GitHub repository <https://github.com/isalykke/NLPexam>

## Pipeline

Below we describe the steps used in finding and analyzing the dataset:

1. Data was scraped from the website [www.grc.com/securitynow.htm](http://www.grc.com/securitynow.htm) using the script ‘*security\_now\_scraper\_beautifulsoup.py*’. The script utilized the **python** package **beautiful soup** (Guido & Drake Jr, 2016; Richardson, 2007).
2. All files were merged and metadata was extracted using the ‘*txts\_to\_csvs.Rmd*’ script. This step was done in **R** (R Core Team, 2019) and implemented in **RStudio** (RStudio Team, 2019).
3. Text cleanup, word removal and lemmatization was done in Visual Studio Codeusing the python package **nltk** (Bird, Klein, & Loper, 2009).
4. LDA topic modelling was similarly done with python utilizing the **genism** package (Radim & Sojka, 2010). The procedure for cleanup and LDA modelling of the data can be seen in the script ‘*one\_lda\_to\_rule\_them\_all.py*’.
5. Google data for comparisons was downloaded from <https://trends.google.com/trends/>. Search parameters were ‘USA’, ‘2004-present’, ‘All categories’ and ‘Web search’.
6. Finally, visualizations of the models were done in both python and R using **ggplot** (Wickham, 2016), **PyLDAvis** (Sievert & Shirley, 2014) and **matplotlib** (Hunter, 2007). These can be found in the scripts ‘*one\_lda\_to\_rule\_them\_all.py*’ (bottom part) and ‘*visualization.Rmd*’.

## Model Specifications

32 LDA topic models were performed – one for each combination of the parameters - in the iterative process described below.

### Cutoffs

After stop word removal, word frequencies were inspected manually to determine an appropriate cutoff for the model. According to Ziph’s law, words that occur more often contain less information (Newman, 2005). As a consequence, the LDA model was run with 3 different cutoffs:

1. A cutoff which removed all words more frequent than the word “windows”. This was the first word determined by the researcher to be informative. Doing so corresponded to removing the 0,22% most frequent words.
2. A cutoff that removed the 10% most frequent words
3. A cutoff that removed the 18% most frequent words
4. A cutoff that removed the 25% most frequent words

### Number of Topics

The optimal number of topics similarly was determined by an iterative process. The model was run with 5, 10, 20, 25, 30, 50, 100, and 150 topics respectively.

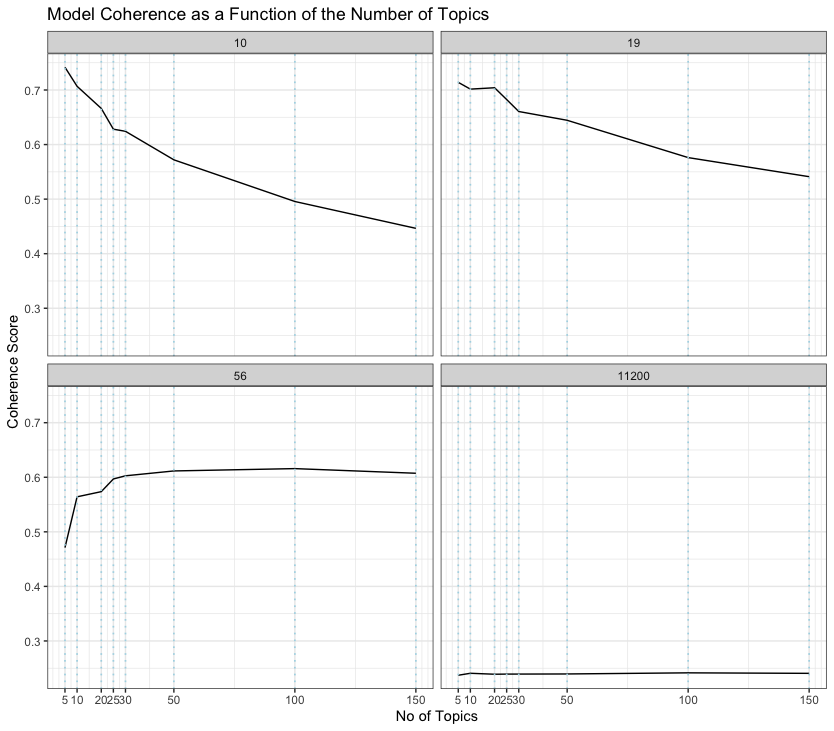
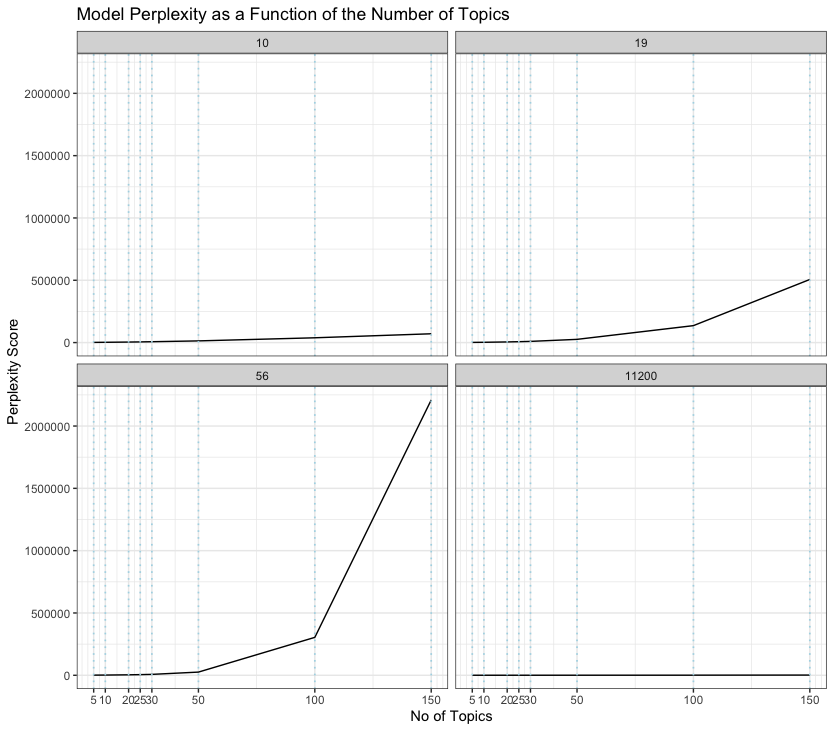
The cutoff was chosen which, combined with the optimal number of topics (see below), resulted in the most promising topic categories, determined by manual inspection. Results from this “optimal model” are reported on in the Results section.

## Model Selection

The best model was chosen based on a mixture of model coherence scores, perplexity scores and manual inspection of the topics. When evaluating the LDA model, there is an obvious tradeoff between model *coherence*, which is a measure of how “good” the learned topics are, and model *perplexity*, which is a measure of how well the model predicts the sample data. Evaluation based on any of these two measures in isolation will likely lead to sub-optimal results (Ding, Nallapati, & Xiang, 2018).

Coherence and perplexity scores for the different parameter combinations can be seen in Fig 2 below as well as in Table A1 in the Appendix.

Figure 2. Coherence- (left) and perplexity scores (right) as a function of the number of topics. In an optimal model, coherence should be high and perplexity low. The four lines visualize the effect of cutoffs removing the 25% (10), 18% (19), 10% (56) and 0,22% (11200) most common words, respectively.



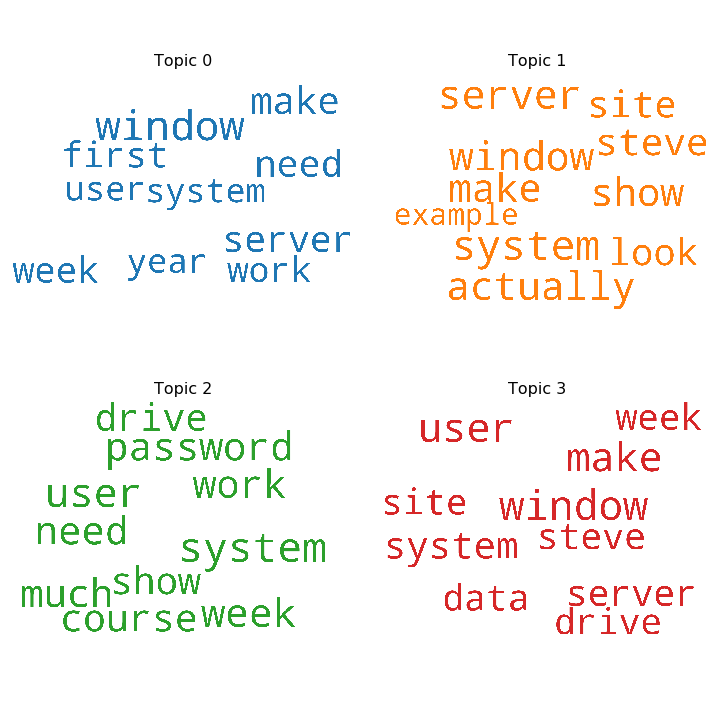
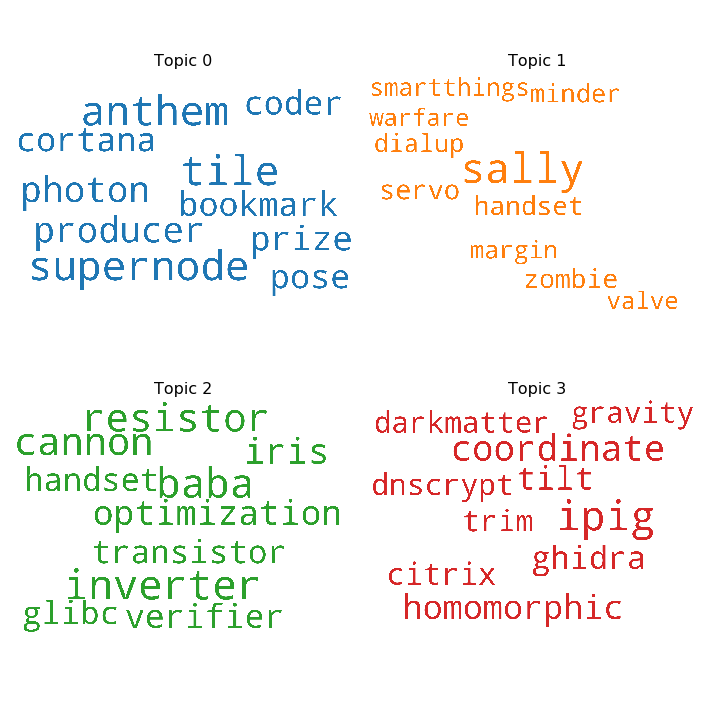
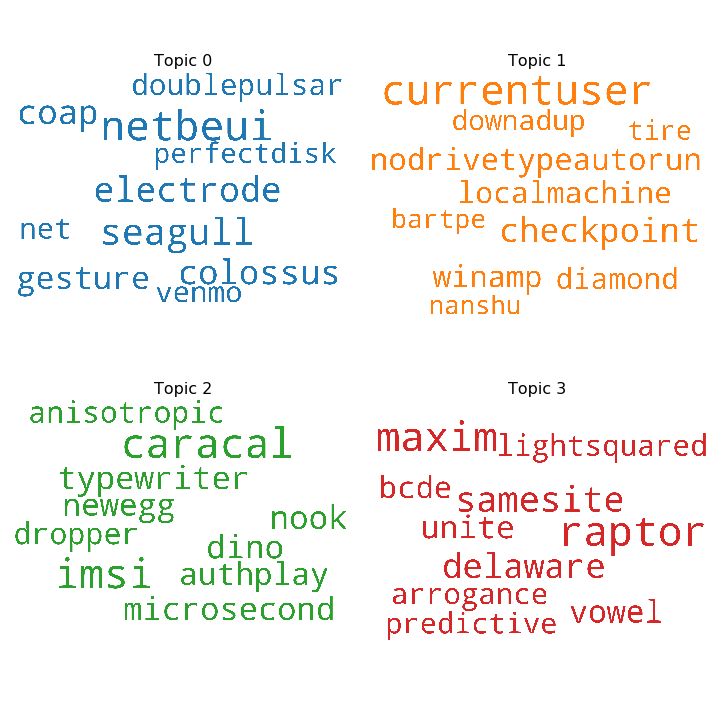
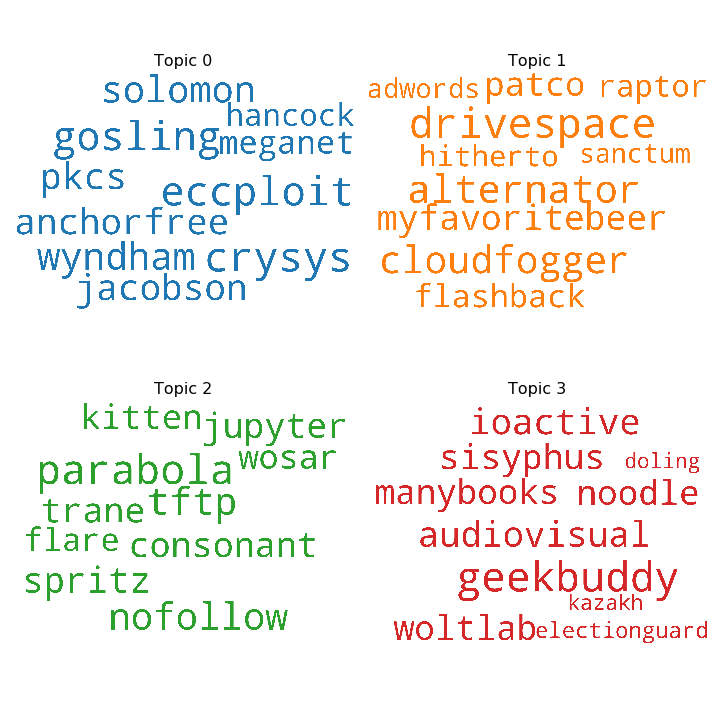
Inspection of Fig. 2 reveals that both cutoff and number of topics are important factors in achieving high coherence, and the same goes for perplexity (except in the case of the 0,22% cutoff, which performs equally bad regardless of the number of topics).

Puzzlingly, perplexity seems to increase as a function of the number of topics, which goes contrary to the expected behavior of the model fitting. We suspect something went wrong in the calculation of this metric, as LDA is supposed to optimize for exactly this feature (‘Perplexity in gensim – Google Grupper’, n.d.). We therefor base our model selection decision mostly on the coherence score.

The effect of different cutoffs on the informativeness of the topic categories can be gauged from Fig. 3. A similar plot for the effect of the number of topics can be found in Fig A1 in the Appendix.

Since topics are created in an unsupervised manner it is not necessarily the case that the model with the highest coherence is also interpretable by humans. Therefor a few of the most promising models were selected and their topics inspected manually.

Figure 3. Word clouds with the same number of topics (10) but different cutoffs. For clarity, only 4 topics per model are displayed. From left to right, top to bottom: 25%, 10%, 18% and 0.22% cutoffs, respectively. Notice how, as the cutoff lowers, the topics become less meaningful, as more “filler-words” are included.



In the end, we wanted a model that could explain potentially many changes over a large timespan, so we choose the model with **100 topics** and choose **a cutoff of 19** (18%) to maximize coherence (see Fig.2).

# Results

## Example Trends: Deepfakes and Tinder

|  |  |
| --- | --- |
| Topic | Keywords |
| 7 | gorn, semicolon, ejbca, flywheel, **deepfakes**, electrode, octet, bros, truste, pono |
| 25 | tcpdf, deserializer, hydro, weblogic, **deepfakes**, cypherpunks, vupen, supercaps, cone, deathswitch |
| 93 | palantir, **deepfakes**, msse, anydvd, netcraft, snob, thrilling, algebra, irfan, rus |
| 34 | **tinder**, acorn, crossrat, authplay, disposeamail, goatse, variablelength, instability, ettercap, vixie |

Et billede, der indeholder tekst

Automatisk genereret beskrivelse

Et billede, der indeholder tekst, kort

Automatisk genereret beskrivelse

# Discussion

### Non-informative Words??

In order to truly know if the podcasts were able to “predict the future” we would have to define a baseline. This is tricky in this case, but theoretically….

## Questions for further research

This dataset still contains many possibilities for further research. Due to time and computing power restraints, we were not able to include these here. Below we mention a number of things that could be interesting to investigate further in future research.

### Bigrams

Future investigations of the dataset should strive to include bigram models in the analysis. Here we mainly investigated the change of specific words over time. Implementing bigram models would allow the researcher to include phrases such as ‘crypto currency, ‘ad targeting’, and ‘SIM jacking’ in the analysis.

### Optimizing of the Model Parameters

Due to time and computational restraints we were only able to test 32 combinations of parameters for the model. Researches with access to more powerful computers might consider adding even more parameter combinations to arrive at the optimal model parameters.

Fig 2 suggests that adding more granulation between the 10 and 19 cutoffs as well as more topic combinations in the range [50-100] might help pin-point the most meaningful model.

### Inspecting Topics

Inspecting and understanding all topics of a large LDA, such as the one presented here is potentially an endless task. However, in the future one should strive to

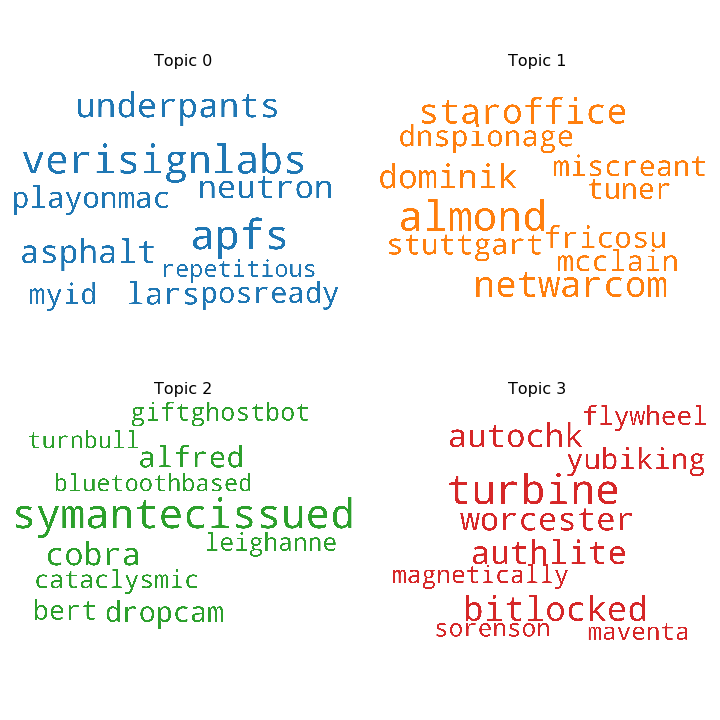
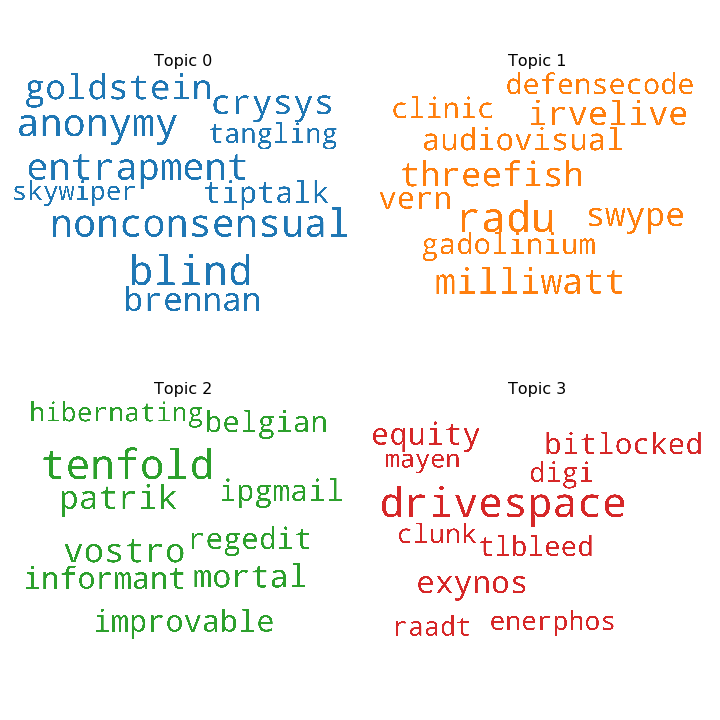
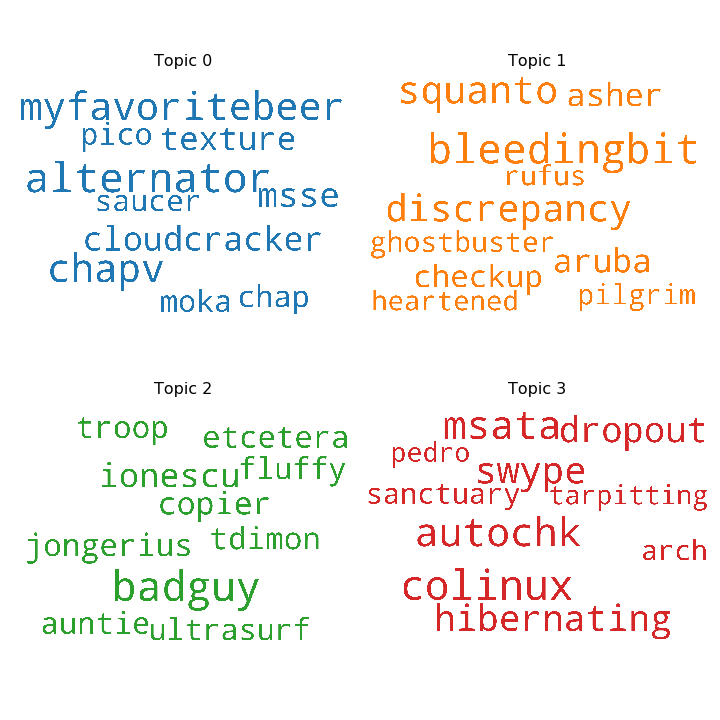
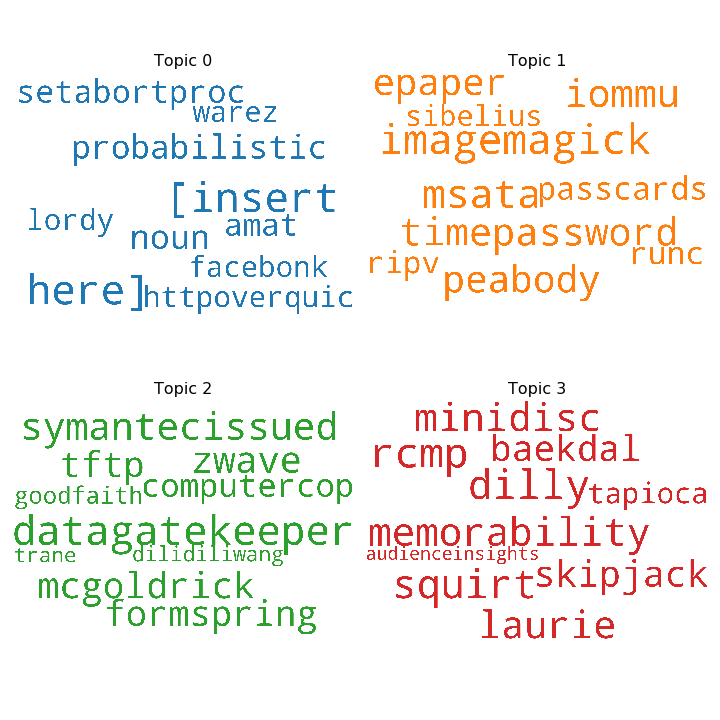
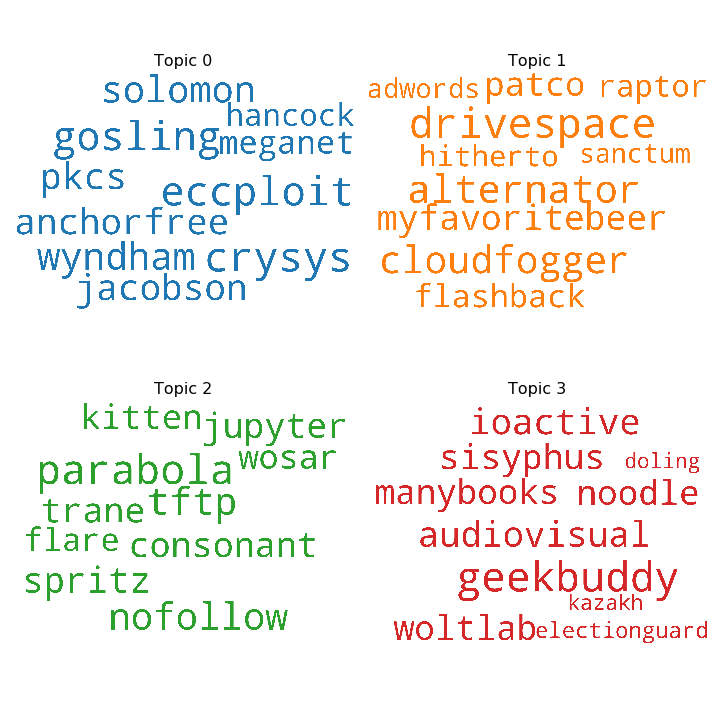
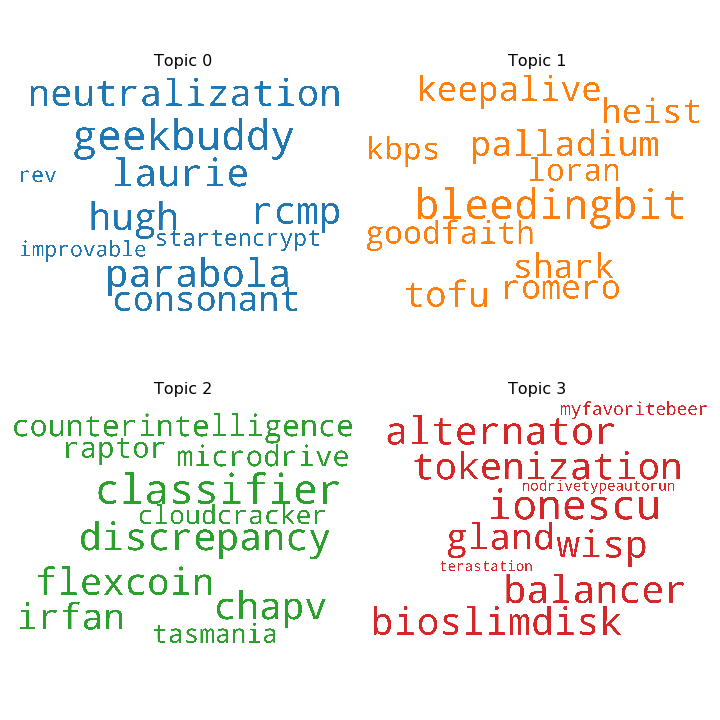
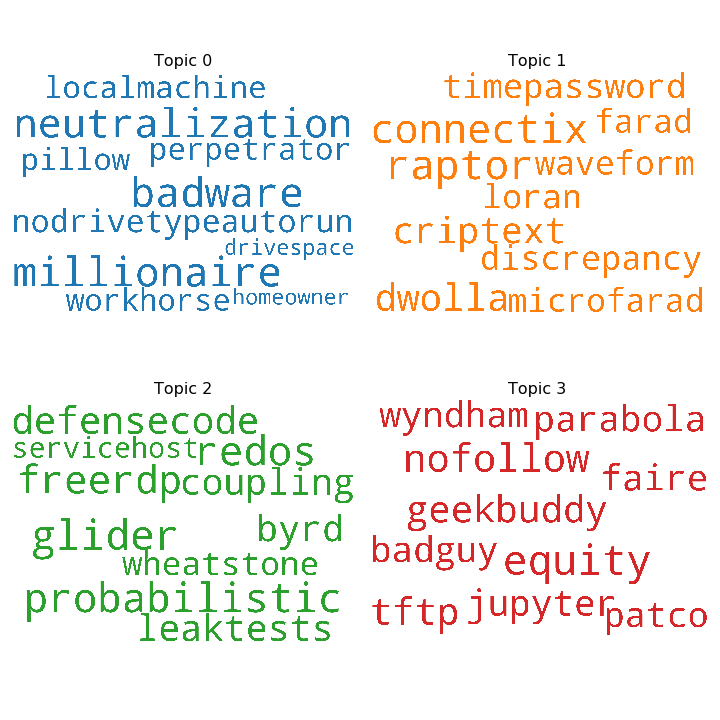
# Conclusion

# References

# Appendix

Table A1. Coherence and perplexity scores for the 32 parameter combinations. Cutoffs correspond to removing 25% (10), 18% (19), 10% (56) and 0.22% (11200) of the most frequent words, respectively.

|  |  |  |  |
| --- | --- | --- | --- |
| No. Topics | CUtoff | Coherence | Perplexity |
| 5 | 10 | 0.7417 | 1514.2217 |
| 5 | 19 | 0.7140 | 1463.5359 |
| 5 | 56 | 0.4711 | 1191.2818 |
| 5 | 11200 | 0.2372 | 275.3456 |
| 10 | 10 | 0.7068 | 2286.5378 |
| 10 | 19 | 0.7016 | 2340.7074 |
| 10 | 56 | 0.5642 | 1734.7232 |
| 10 | 11200 | 0.2408 | 292.75245 |
| 20 | 10 | 0.6662 | 4175.6955 |
| 20 | 19 | 0.7041 | 5127.4933 |
| 20 | 56 | 0.5736 | 3695.9150 |
| 20 | 11200 | 0.2390 | 334.1970 |
| 25 | 10 | 0.6282 | 5399.2013 |
| 25 | 19 | 0.6825 | 7141.7019 |
| 25 | 56 | 0.5969 | 5274.3530 |
| 25 | 11200 | 0.2393 | 358.2465 |
| 30 | 10 | 0.6243 | 6924.9440 |
| 30 | 19 | 0.6606 | 9601.0081 |
| 30 | 56 | 0.6027 | 7311.9951 |
| 30 | 11200 | 0.2393 | 381.2567 |
| 50 | 10 | 0.5720 | 13559.0001 |
| 50 | 19 | 0.6445 | 25887.0620 |
| 50 | 56 | 0.6116 | 25586.2148 |
| 50 | 11200 | 0.2395 | 498.3244 |
| 100 | 10 | 0.4957 | 38494.5352 |
| 100 | 19 | 0.5761 | 136020.7153 |
| 100 | 56 | 0.6159 | 304331.2374 |
| 100 | 11200 | 0.2416 | 938.1294 |
| 150 | 10 | 0.4465 | 70763.2530 |
| 150 | 19 | 0.5411 | 506522.2001 |
| 150 | 56 | 0.6073 | 2209408.7959 |
| 150 | 11200 | 0.2406 | 1705.6233 |



A1. Word clouds with the same cutoff (10/25%) but different number of topics. For clarity, only 4 topics per model are displayed. From left to right, top to bottom: 5, 10, 20, 25, 30, 50, 100, and 150 topics respectively. Notice how, when you allow the number of topics to go high enough, even the word “underpants” merits its own topic.

1. After the German mathematician Peter Gustav Lejeune Dirichlet [↑](#footnote-ref-1)