Natural Language Processing for Air Force Policy Research

Gavin S. Hartnett*
RAND Corporation, 1776 Main St., Santa Monica, CA 90401

In this Technical Note I describe how web-scraping techniques were used to compile a corpus of roughly 9000 official US government policy and doctrine documents at the DoD, Joint, and AF-levels. This corpus can be used to facilitate policy research when combined with Natural Language Processing techniques. To this end, I will introduce a simple search tool based on substring searches, as well as more complicated search methods based on document embeddings using the Doc2Vec algorithm. The quality of the search results indicates that the document embeddings have truly encoded semantically meaningful information. Both the data corpus and the techniques I discuss should be broadly useful across many areas of defense policy research.

First published internally within RAND on December 19, 2019.

INTRODUCTION

A significant challenge facing national security policy researchers is the large volume of official doctrine and policy documents. The US DoD is often described as the largest bureaucracy in the world, and it produces a proportionate amount of guidelines and formal documentation. For some policy research problems, especially those which are narrow in scope, there will likely be just a few key references, and the large volume of documents will not pose a significant challenge. However, it is often the case that some of the most difficult and important policy problems are those that cut across many domains and involve many different stakeholders. For these problems, the fact that it is impossible for any one person to read even a fraction of the relevant documents in a timely manner poses a real constraint.

In both academia and industry, machine learning and data science more generally are increasingly being used to leverage vast quantities of data into useful insights and analysis. Data science comprises an incredibly broad collection of techniques and methodologies, and there are many ways in which these could be brought to bear on policy problems. Often, the limiting factor is a good source of data, but in the case of analyzing DoD policy, the vast quantity of textual data is both an opportunity and a challenge that must be confronted, and it is therefore rather natural to apply recent advances in natural language processing (NLP) in order to aid policy researchers by quickly and accurately synthesizing large quantities of text.

In this Technical Note I report some recent progress towards this goal for the specific case of Air Force policy research, with an emphasis on problems which cut across many areas within the DoD - particularly at the joint level.

CORPUS GENERATION

A corpus of official and current policy documents was assembled by drawing from all documents hosted on these four US government websites:

1. Joint Chiefs of Staff Joint Doctrine Pubs[1]
2. Curtis E. LeMay Center for Doctrine Development and Education[2]
3. DoD Directives Division Issuances Program[3]
4. AF E-Publishing[4]

I specifically restricted attention to documents obtainable through official government websites, which ensured that the documents did in fact represent current official policy. I also considered only websites which were publicly-available and thus unclassified. It would be interesting to extend the techniques discussed here to classified settings, although this would clearly introduce many additional complications. Each source contained many different types of documents. For example, the DoD Directives website contained Directives, Instructions, Manuals, Administrative Instructions, and Directive-Type Memoranda.

The above four sources together host thousands of documents - far too many to be manually downloaded. Therefore, I used the software library *Selenium* to crawl the websites and automatically download every accessible document.[5] This resulted in a corpus of roughly 9000 documents in pdf file format. An additional benefit of automating the downloading is that the corpus may be

[1] https://www.jcs.mil/Doctrine/Joint-Doctine-Pubs/
[2] https://www.doctrine.af.mil/
[3] https://www.esd.whs.mil/dd/
[4] https://www.e-publishing.af.mil/Product-Index/
[5] Not all documents listed on the sites were able to be downloaded, however; some were restricted and required additional action in order to be accessed.

quickly and easily updated. In the course of the project I noticed that the online databases had documents added, removed, or changed on a weekly basis.

In order to facilitate textual analysis, the pdfs were then converted to raw text using the command-line utility "pdftotext". Visual inspection of a small random selection of documents (both the original PDF and the generated raw text file) confirmed that this command works well and yields good raw-text representations of the original PDF document. One complication I encountered was that some of the pdfs were password-protected which prevented pdftotext from working. To surmount this difficulty, an object character recognition (OCR) tool could have been used, but I chose to simply ignore these documents.

CORPUS DETAILS

Next I will describe some basic details and statistics of the corpus. All documents in the corpus were downloaded between Sept. 8-10, 2019. In the end, I obtained 8026 total documents in raw text form, taking up a total of 700 Mb of disk space.[6] 8026 is clearly a very large number of documents from the perspective of a human researcher, but it is not large by modern NLP standards. However, it is also the case that many of the documents are quite large, some numbering in the hundreds of pages. More insight into the size of the corpus may be obtained by plotting the distribution of word counts, which is done in Fig. 1 (a), (b). Note that in these plots the word count is derived from the raw text conversions of the downloaded pdf documents. Thus, no processing has been done to remove erroneous words.[7] Overall, the total number of words is about 101M, and the number of distinct words (i.e. the size of the vocabulary) is 1.5M. I stress that both of these counts are inflated due to the presence of erroneous words and thus these statistics represent a raw measure of the data before any pre-processing has been performed.[8]

To gain more insight into the distribution of words in this corpus, in Fig. 1 (c) I plot the rank vs. frequency for each word in the corpus (the rank is defined such that the most common word has rank 1, the second most common has rank 2, etc). The data roughly follows a Zipfian distribution with exponent $s = -1.34$. Interestingly, the data seems to exhibit separate power-law regimes for word rank $\lesssim 1000$ and word rank $\gtrsim 1000$.

SUBSTRING SEARCH

Simply obtaining a corpus of official documents of this size represents a significant resource for policy researchers. In particular, access to such a corpus immediately lends itself to a simple keyword search functionality based on substring matching. This could be used to return all the documents which mention specific terms, for example "STRIKWARN". Depending on how common the term is, this simple search could be used to very quickly retrieve documents relevant for a particular policy question.

An additional benefit of this search functionality is obtained when the names of other documents in the corpus are used as the keywords in the search. This is aided by the fact that many official US defense policy documents have systematic naming conventions - for example the DoD Instruction entitled *The Chemical, Biological, Radiological, and Nuclear (CBRN) Survivability Policy* is named DoDI 3150.09. Any specific reference to the substring "DoDI 3150.09" indicates that that document is citing this DoD Instruction, and moreover, any hits to the more inclusive search "3150.09" are also likely referencing this Instruction. This functionality may be used on a case-by-case basis (i.e. it allows one to answer the question "Which documents in the corpus cite document X?"), but in principle it could also be applied to the entire corpus in order to infer the citation graph. This would be a useful direction for future work.

DOCUMENT EMBEDDINGS

It is useful for many NLP tasks to have a condensed representation of the textual data. One way of obtaining such a representation is to use document embeddings, and to this end I used the *Gensim* implementation of

[6] A larger number of documents (8649) were downloaded and converted to raw text. Unfortunately, some documents are restricted and cannot be downloaded, even though they are listed on the web-page. In some cases, restricted documents cause the download operation to fail, while in others, the download superficially succeeds, but the resulting pdf file is either blank or contains a short statement such as "STOCKED AND ISSUED". Unfortunately, the placeholder pdf for restricted documents is not standardized, making the task of filtering out these documents non-trivial. I adopted a simple approach of simply removing all documents with less than 100 words.

[7] For example, *pdftotext* often converts the page numbers and section numberings into text strings separated from the main body by spaces, and thus these would count as words under our approach.

[8] In particular, by most counts the number of distinct English words is $\lesssim 200k$.

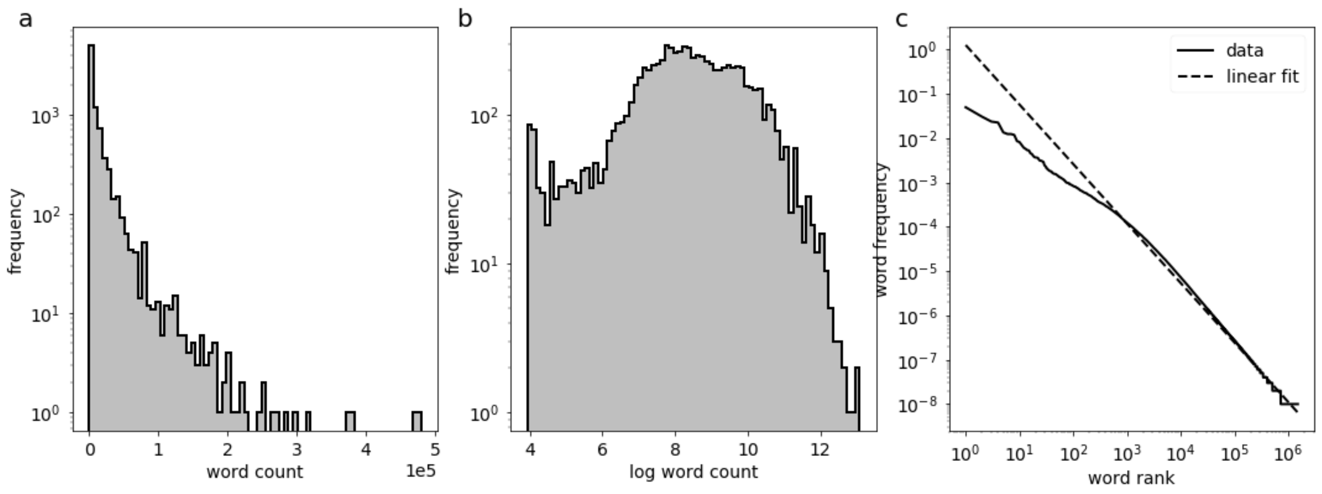

FIG. 1: (a) The distribution of document word counts for the corpus. (b) The distribution of document log word counts for the corpus. (c) Word rank vs. word frequency, plotted on a log-log scale. A linear fit to the log-transformed data (dashed line) yields a Zipf-exponent of -1.34.

Doc2Vec [2] (itself based on the Word2Vec word embedding [3]) to map each document in our corpus to a d-dimensional real-valued vector. I then used this mapping for numerous tasks, but first, I will describe the implementation details.

I obtained good results with $d = 100$, although it would be interesting to investigate a range of dimensions. Doc2Vec was trained for 100 epochs, starting with an initial value of $\alpha = 0.025$ which was then decreased by 0.0002 after every epoch. The Gensim hyperparameter `dm` was set to `dm`= 1, corresponding to a distributed memory approach, and I also set `min_count` = 10, so that all words which appear less than 10 times in the corpus are ignored. In terms of data pre-processing and formatting, all text was converted to lower-case, all numbers and stop words were removed, and the documents were lemmatized using the NLTK WordNet lemmatizer. Doc2Vec uses categorical tags for the documents. I used the following tags:

Tags:
`AF, DoD, Lemay Center, JCS, AF:bases, AF:departmental, AF:dru, AF:foa, AF:majcom, AF:natlguard, AF:numberedAFB, AF:units, DoD:admin_instructions, DoD:directives, DoD:dtms, DoD:instructions, DoD:manuals, JCS:joint_doctrine_pubs`

Lastly, I chose to apply Doc2Vec to each individual document. An alternative approach would be to apply it to sub-components of documents, for example to individual sections, sub-sections, or even paragraphs. Applying Doc2Vec at this level of granularity is appealing because some of these documents are hundreds of pages long and encompass a broad collection of topics, but my focus was on quickly searching through the corpus at the document level, and with this goal in mind it seemed reasonable to apply Doc2Vec at the document level.

A major benefit of document embeddings is that the representations naturally induce a distance metric between documents. Denoting the vector representation of document i as $\boldsymbol{v}^{(i)}$, then the cosine similarity between documents i and j is defined as

$$S(\boldsymbol{v}^{(i)}, \boldsymbol{v}^{(j)}) = \frac{\boldsymbol{v}^{(i)} \cdot \boldsymbol{v}^{(j)}}{||\boldsymbol{v}^{(i)}||_2 ||\boldsymbol{v}^{(j)}||_2} , \qquad (1)$$

where $\cdot$ is the vector dot product and $||\boldsymbol{v}^{(i)}||_2$ is the Euclidean ℓ_2 norm of the vector. The cosine similarity is 1 for two parallel vectors, and -1 for anti-parallel vectors. Additionally, the angular distance may be defined as

$$D(\boldsymbol{v}^{(i)}, \boldsymbol{v}^{(j)}) = \frac{1}{\pi} \cos^{-1}\left(S(\boldsymbol{v}^{(i)}, \boldsymbol{v}^{(j)}) \right) . \qquad (2)$$

$D \in [0, 1]$, with the minimal value obtained for parallel vectors and the maximal value obtained for anti-parallel vectors. By evaluating the angular distance for every distinct pair of documents, one may obtain a distance matrix with matrix elements $d_{ij} = D(\boldsymbol{v}^{(i)}, \boldsymbol{v}^{(j)})$.

Nearest Documents

The distance matrix may be used for many additional tasks. In particular, it enables a second type of search method. If a policy researcher has identified a particular document as being relevant, and they are interested in seeing if there are other documents on the same topic that they should be aware of, one approach would be to use the distance matrix to find the closest documents to the reference one. In particular, if the reference document is i, then $\min_{j\neq i} d_{ij}$ will be the closest document in terms of angular distance. Of course, there is no absolute or true intrinsic way to measure the distance between documents, and the true test of the angular distance measure induced by the vector representations will be whether it is useful or not. For example, in Table I I show a few examples of the nearest documents for a given reference document chosen mostly arbitrarily from the corpus.

The relevance of the nearest documents to the reference one is impressive - although in most cases the titles have overlapping keywords. All documents in the 1st row have the words comprising the CBRN acronym in the title (chemical, biological, radiological, and nuclear), all documents in the 3rd row have "Depot Maintenance", and all documents in the 4th row have either awards or recognition. Importantly, the Doc2Vec algorithm did not have access to the titles beyond the fact that the title of a document often appears near the beginning of the converted raw text file. The results of Table I indicate that the vector representations have successfully encoded semantic meaning, but the fact that the titles of each document are so similar suggests that the same effect could have been achieved by keyword searches restricted to the titles.

Hierarchical Clustering

Rather than simply searching through the sorted list of document distances d_{ij} for some reference document i in order to find the closest documents, a more sophisticated approach would be to apply a hierarchical clustering algorithm, which grows a binary tree using the documents as leaf nodes. The algorithm then iteratively builds clusters of documents. Initially, each document represents its own cluster, and then at each step in the algorithm a new cluster is formed which represents the two closest clusters. The algorithm terminates when there is just a single cluster, corresponding to the root node in tree. Thus, the number of clusters is a function of the height of the tree, and ranges from 1 at the base of the tree to the number of documents in the corpus at the leaves.

I applied the scikit-learn implementation of this algorithm[9] to the entire corpus, using the distances d_{ij} and the "single" option, corresponding to the Nearest Point Algorithm. An example for a random selection of just 50 documents is shown in Fig. 2 as the tree for the full corpus is quite large. The full dendrogram is shown in

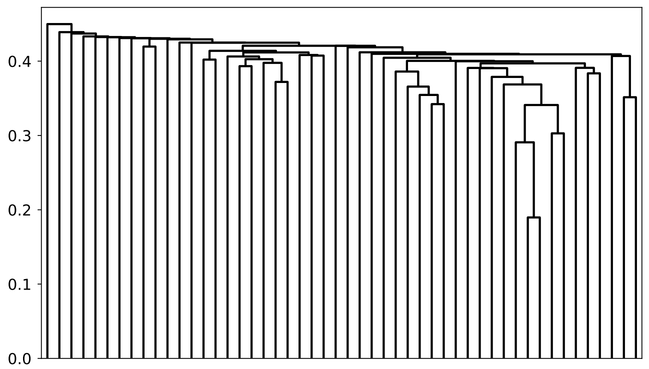

FIG. 2: The binary tree (sometimes called dendrogram graph) representing the result of the hierarchical clustering algorithm for a random selection of 50 documents out of the corpus.

Fig. 3, together with a heatmap of the distance matrix d_{ij} for all 8026 documents in the corpus.

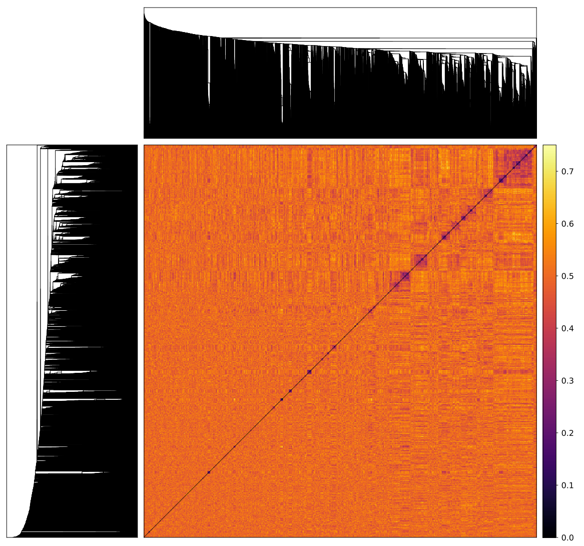

FIG. 3: The full corpus dendrogram, plotted together with a heatmap for the distance matrix d_{ij}. The indices of the matrix have been permuted so that they align with the leaves of the dendrogram diagram. Thus, tight clusters in the dendrogram are associated with the darker colored square regions within the heatmap. The diagonal strip in the heatmap corresponds to the $i = j$ entries in the matrix, for which $d_{ii} = 0$. Note that the matrix has been rotated to conform to the ordering of the leaves.

Cutting the tree at some prescribed height yields a collection of clusters of different sizes. These might be useful

[9] https://docs.scipy.org/doc/scipy/reference/generated/scipy.cluster.hierarchy.linkage.html

Reference Document	Nearest Document	2nd Nearest Document
AFTTP 3-2.55 *Chemical, Biological, Radiological, and Nuclear Threats and Hazards*	AFTTP 3-2.46 *Multi-Serice Tactics, Techniques, and Procedures for Chemical, Biological, Radiological, and Nuclear Passive Defense*	AFTTP 3-2.42 *Multi-Service Doctrine for Chemical, Biological, Radiological, and Nuclear Operations*
DoD Directive 4650.05 *Positioning, Navigation, and Timing (PNT)*	DoD Directive 5100.96 *DoD Space Enterprise Governance and Principal DoD Space Advisor (PDSA)*	DoD Instruction 4650.08 *Positioning, Navigation, and Timing (PNT) and Navigation Warfare (NAVWAR)*
DoD 4151.18-H *Depot Maintenance Capacity and Utilization Measurement Handbook*	DoD Instruction 4151.20 *Depot Maintenance Core Capabilities Determination Process*	Air Force Sustainment Center (AFSC) Guidance Memorandum (GM) to AFSCMAN 21-102 *Depot Maintenance Management*
ACC Instruction 36-2801 *Awards and Recognition Programs*	437 Airlift Wing Instruction 36-2805 *Wing Recognition Program*	51st Fighter Wing Instruction 36-2805 *Quarterly and Annual Awards Program*

TABLE I: The nearest, and 2nd nearest documents for a given reference document.

in constructing groupings of similar documents. There does not appear to be a natural choice where to make such a cut - in the current setting the choice should be driven by the usefulness of the result. Fig. 4 (a)-(f) shows results for 3 different choices of cuts. Each cut produces a different number of clusters, N_c, which ranges from 1 to N, the corpus size. Fig. 4 (a) shows the extreme case where each document forms its own cluster, corresponding to cutting the tree at the leaf nodes. As the height of the cut moves closer to the root node, the clusters start grow by merging, so that the overall number of clusters decreases. Fig. 4 (f) shows the result of a cut very close to the root node, where there are very few clusters. Interestingly, many of the documents are still distributed among very small clusters, which indicates that the corpus consists of many very distinct documents and a few very similar ones (with distance measured in terms of the angular distance induced by the document embeddings). This general trend is also depicted in Fig. 2 and Fig. 2, which both show that there are still many clusters at a height close to the root node.

The hierarchical clustering can also be used to search for similar documents in the following manner. Rather than using the distance matrix d_{ij} to return to K-nearest documents, the clustering assignments encoded in the binary tree could be used to return similar documents to a reference document. For example, in Fig. 5, starting with the reference document DoD Directive 4650.05 *Positioning, Navigation, and Timing (PNT)*, I find that it is assigned to the same cluster as 3 other documents:

- DoD Directive 3100.10
  Space Policy
- DoD Directive 5100.96
  DoD Space Enterprise Governance and Principal DoD Space Advisor (PDSA)
- DoD Directive 3100.16
  DoD Management of Space Professional Development

To broaden the search, this process could be repeated to find the cluster that joins the current cluster of all 4 documents (and so on). This approach is not equivalent to the K-nearest neighbor search discussed above, as evidenced by the fact that the 2nd closest document found in Table I is not a member of this cluster.

CONCLUSIONS AND OUTLOOK

I used simple web-scraping techniques to compile a corpus of roughly 9000 official US government policy and doctrine documents at the DoD, Joint, and Air Force-levels. I showed how this corpus could be used for policy research through simple substring searches, and I also used Doc2Vec to build document embeddings which map each policy document to a real-valued vector. These em-

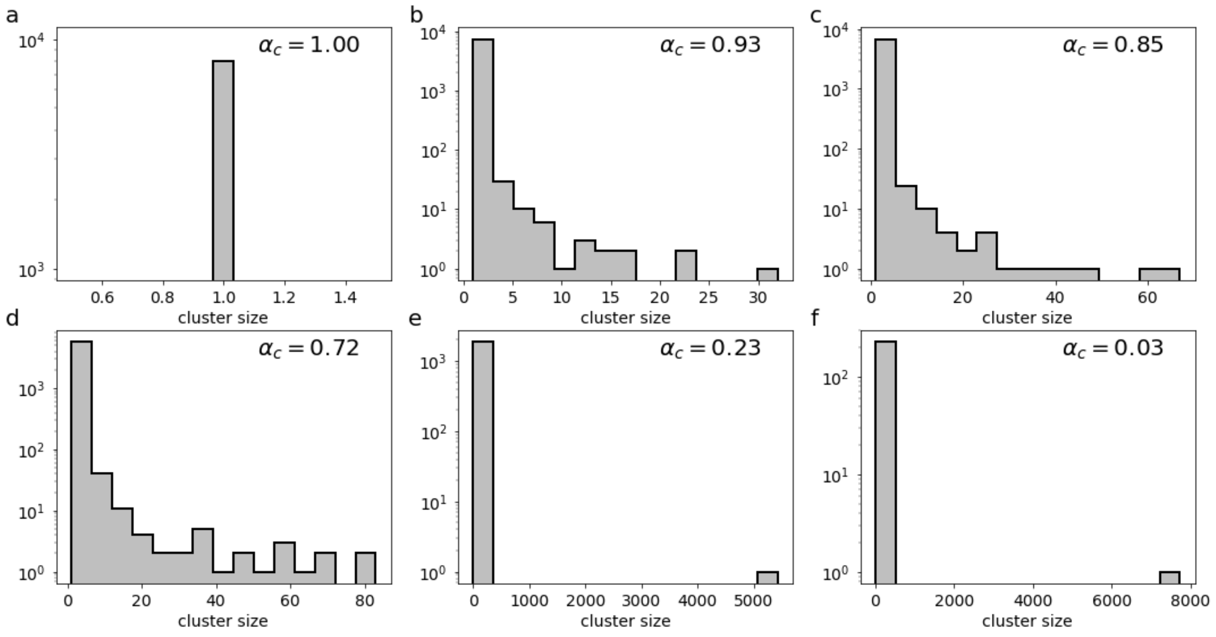

FIG. 4: The distribution of cluster sizes for various cuts of the dendrogram/binary tree. Here $\alpha_c := (N_c - 1)/(N - 1)$ represents the fraction of possible clusters. A value of $\alpha_c = 1$ corresponds to the case when there are N clusters, so that each document belongs to its own cluster. Conversely, $\alpha_c = 0$ corresponds to the case when there is a single cluser containing the entire corpus.

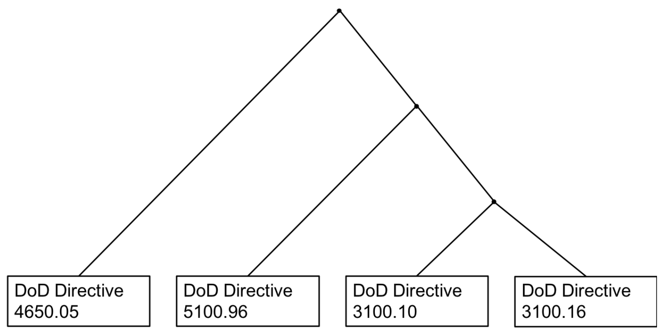

FIG. 5: The sub-tree associated with the sub-cluster of 4 DoD documents discussed in the text.

beddings can be used to quantify the similarity of documents to one another, and this was used to develop additional search techniques. For many of NLP methods, including those discussed here, there is no absolute performance metric that can be used for evaluation. Ultimately then, evaluation is determined according to whether the method was useful or yielded some interesting new insight.

The RAND project for which this work was carried out has ended, but there are many interesting extensions and new directions which could be developed in future work. From a methodological standpoint, I only explored a small subset of the possible NLP techniques that could have been applied to this corpus. For example, topic modeling through Latent Dirichlet Allocation [1] is commonly used to gain a sense of the topics contained within a corpus. More interestingly, there have been major advances in NLP and language modeling using so-called transformer models (see [5] for a review) based on the attention mechanism [4]. It would be interesting to apply these state-of-the-art techniques to the problem of analyzing defense policy documents.

Acknowledgements: It is a pleasure to thank Don Snyder for leading the PAF project under which this work was carried out, as well as Kurt Klein, Jake McKeon, John Speed Meyers, and Jim Powers for their collaboration on this project. I would also like to thank Dara Gold for introducing me to *Selenium* and sharing her code with me, and Drew Lohn and Edward Geist for reading a draft of this Note and providing me with useful feedback. Lastly, I would like to thank the FMEP leadership team Jim Chow and Elizabeth Bodine-Baron for supporting the publication of this note.

* hartnett@rand.org

[1] Blei, D. M., Ng, A. Y., and Jordan, M. I. Latent dirichlet allocation. *Journal of machine Learning research 3*, Jan (2003), 993–1022.

[2] Le, Q., and Mikolov, T. Distributed representations of sentences and documents. In *International conference on machine learning* (2014), pp. 1188–1196.

[3] Mikolov, T., Chen, K., Corrado, G., and Dean, J. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).

[4] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *Advances in neural information processing systems* (2017), pp. 5998–6008.

[5] Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., et al. Transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771* (2019).