What's Happening in AI!? Hierarchical Topic Analysis for Artificial Intelligence Tweets

David Koleczek
University of Massachusetts Amherst
dkoleczek@umass.edu

Abstract

The goal of this project is to distill the breadth of knowledge and news about AI, machine learning, and data science that is shared on Twitter. In a previous project we took one step towards this goal by classifying tweets based on their relevance to AI and then creating a bot[1] *to retweet the most relevant tweets. In this project we aim to create an unsupervised system to discover high level topics, repeat to discover sub-topics, and finally calculate their relative frequency over time with respect to the other topics.*

1. Introduction

Artificial intelligence, data science, and machine learning (from this point we will refer to these related fields as AI) is a field that has become incredibly vast in the amount of different subfields and practical problems it is being applied to. For a data scientist in industry or a researcher in computer science, keeping up with the latest research, trends, and applications can be a daunting task. While in parallel social media like Twitter has made it easier than ever to share information across institutions and disciplines; it still faces several issues that make it difficult to use for this task out of the box. It has a lack of focus in that content recommendations and discovery are not tailored to this task, rather will provide recommendations to suit the goals of the platform. It is rare that a user would only like to see AI content on their Twitter feed all of the time and would instead prefer to purposefully seek this knowledge. Our goal with this work is to leverage the information shared on social media, namely via Twitter, and present it in a more focused digest about AI.

In a previous project, we created a system that collects tweets and scores them based on their relevance to AI . The most relevant tweets were both retweeted by an account on Twitter at a fixed schedule, and also displayed in a feed on a web UI. This system was able to effectively separate relevant tweets from the remaining noise that can be found on social media. However, a drawback is that it finds *all* AI content. While still substantially focusing content that can be seen by a user, it still may be overwhelming to see 20-30 tweets a day ranging from theory to reinforcement learning to natural language processing. It also makes it difficult to understand a holistic picture of trends in the field. For example, someone may currently be seeing many new tweets about a new transformer model, but they would not know this is a dominant paradigm in NLP if they did not consistently check the filtered Twitter feed over time.

As a solution to these problems, we propose an unsupervised method to discover topics within a corpus of AI tweets. These discovered topics would be tracked over time in order to provide a holistic picture of AI trends. Furthermore, the high level topics found will be decomposed into subtopics using a similar unsupervised method. In this way, our goal is to first discover topics such as "image recognition", "reinforcement learning", or "deep learning". Then, of the data that is classified as "reinforcement learning", for example, we will be able to further decompose it into *subtopics* such as "environments" or "multi agent". For each hierarchy of discovered topics, we will chart the relative number of tweets associated with each topic over time as a proxy for the importance of the topic. This is based on the intuition that more important topics during a period of time will have more associated tweets. Finally, at the subtopic level we can view the associated tweets for some particular time period. We envision a use-case in a system where a user will select a high level topic they are interested in, view the subtopics, and select a subtopic which will finally show the tweets associated with that subtopic between some time period.

To solve this problem, we propose a topic modeling approach closely related to BERTopic [5]. We make several key changes and additions to build an end-to-end system for this task. We also provide a detailed analysis of the results and provide discussion on the practical utility of this approach.

[1] Found on Twitter dave_co_dev

1.1. Data

The dataset is a collection of tweets from the home timeline of an account that follows AI-related twitter accounts (individuals in academia or industry, and organizations). There are tweets from any account on Twitter, including ones not followed by the account. Some of the most frequent users are hardmaru, François Chollet, and Yann LeCun. The core of the dataset is the full tweet text, its language, the user name, creation date, and what type of tweet it is (regular, retweet, reply, or quote). The tweets are fetched directly from Twitter's API every 10 minutes and stored in a database. Some tweets contain URLs that provide important context. For each URL we have scraped the website's title and description text (this is what is usually in Twitter's preview) and the full HTML content. Additionally, not every tweet in this dataset is relevant to AI. In a previous project we created a classifier to predict if a tweet is relevant. This model is trained using 9000 ground truth binary labels we have manually assigned through a web UI[2] over the last few years. The model itself is fine-tuned RoBERTa [11] which outputs a relevance score from 0 to 1. We found that tweets with scores > 0.8 can be considered relevant. A total of 27000 relevant tweets, mostly from September 2019 through September 2021, will be used for unsupervised learning.

2. Related Work

2.1. Latent Dirichlet Allocation

One of the most commonly used methods of topic modeling is Latent Dirichlet Allocation (LDA) [2]. LDA is a generative approach to topic modeling that assumes all documents within a corpus belong to some *predetermined* number of topics. This means that the optimal amount of topics for the task must be known ahead of time and each document must be assigned to one of the topics. It learns probabilities to estimate the topic distribution for each document based on the distributions of words within each topic and document. Topics become the words that are the most likely words for the associated topic cluster, and the topic for a document is the one with the highest probability. In LDA each document is treated as a bag of words without regard for the semantics of the overall document. This results in the model being sensitive to the pre-processing of the document (stop words are often filtered out and n-grams are created).

2.2. Embed, Cluster, Extract

Recent work in topic modeling has taken a multi-step approach instead of relying on an individual algorithm like LDA. [1] and [5] use a paradigm that we refer to as *Embed, Cluster, Extract*.

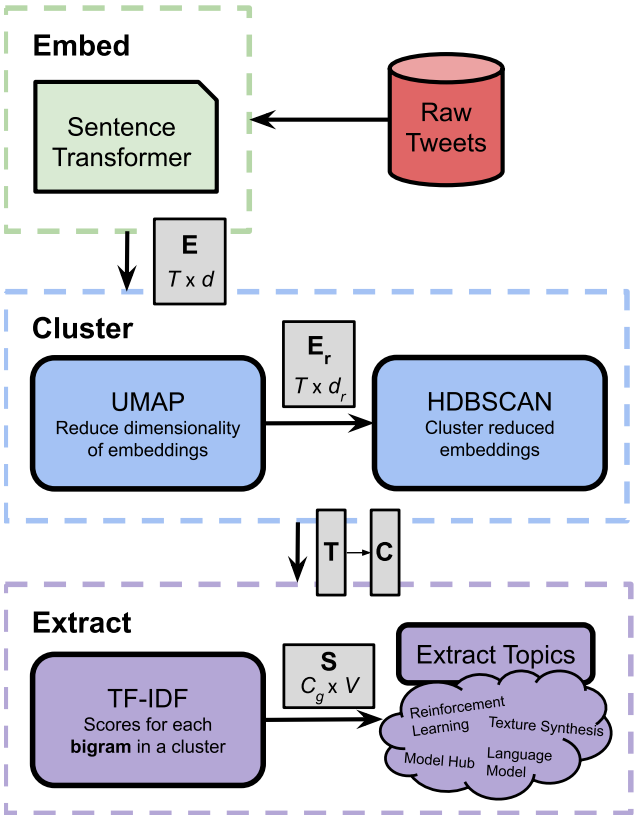

Figure 1: *Embed, Cluster, Extract* paradigm we use for topic modeling based on BERTopic [5].

Embed. Documents are first embedded using pre-trained models such as *doc2vec* [9] or a sentence transformer [19]. Sentence transformers in particular are specifically designed BERT-based [4] models for generating sentence embeddings that are more effective for downstream tasks such as topic modeling than directly using BERT's output layer or a vector representation model like GloVe [15].

Cluster. Given a collection of embedded documents, the next step is to cluster documents, where cluster membership corresponds to a document's topic. The embeddings produced in the previous step by doc2vec generally have a length of 100, while the embeddings produced by a sentence transformer can be up to 768. This high dimensionality poses a problem for clustering algorithms. As such, before clustering the data, it is reduced using an algorithm such as UMAP [13]. Finally, an algorithm such as HDBSCAN [3] is used for creating clusters of documents.

Extract. Once documents are clustered, an interpretable topic description is created. [1] uses the centroid of the cluster and finds the closest word vectors. [5] treats all docu-

[2]Found at mlfeed.tech

ments in a single topic as a single dataset and applies TF-IDF which gives the importance scores for words within a cluster.

Embed, Cluster, Extract is the paradigm our work is based on and we will later provide a discussion about its weaknesses in practice as this is a notable omission from the aforementioned related works.

2.3. Autoencoders

Originally proposed in [8], [21] uses an autoencoder neural network model for topic modeling. The goal of an autoencoder is to reconstruct an input sequence from a compressed representation within the model. Using such an architecture, their autoencoder uses as input document embeddings. The input embeddings were originally GloVe, but [21] used their novel sentence transformer Phrase-BERT. They showed that the resulting topics are more coherent than LDA via human evaluation experiments.

3. Approach

3.1. Baseline

We establish Latent Dirichlet allocation (LDA) [6, 2] as our baseline method of topic modeling. Due to LDA being very sensitive to the tokens contained within the documents take a number of steps to clean the raw tweet data. Leveraging the NLP library spaCy [7], we first apply a number of corpus independent transformations on each document. The transformations are: lowercase each token, remove stop words, remove punctuation, symbols, and URLs, lemmatize tokens, and remove any tokens beginning with '@' (mentions on Twitter). Afterward we apply the following corpus dependent transformations: adding the most common bigrams and trigrams to each document, remove tokens that occur in less than 20 documents, and removing tokens that occur in over 25% of documents. Using gensim [18] we convert each document into a term-frequency inverse document frequency (TF-IDF) [17], bag of words representation. Finally, using these vectors we learn an LDA model.

3.2. Embed, Cluster, Extract

Using BERTopic [5] as a starting point, our primary system for topic modeling uses the *Embed, Cluster, Extract* paradigm we previously described and an overview is shown in Figure 1. In order to categorize tweets into topics, we first need to create a vector representation that represents the text of each tweet. Given an input tweet of length N tokens, we compute a representation x. To compute this representation, we use sentence transformers [19] which use the standard BERT architecture [4] to create contextualized word embeddings for all input tokens. However, since a fixed length output is desired, a mean-pooling operation is applied to BERT's output. We use the pretrained *all-mpnet-base-v2* model based on MPNet [20] as it has the best performance on the evaluated downstream tasks[3] and computational speed is not a concern. We individually embed each tweet into a T x d embedding matrix $\mathbf{E}$ where T is the number of tweets and d is the dimensionality of each embedding.

Once we have $\mathbf{E}$, we need to reduce the dimensionality of these embeddings in order for them to be used effectively in a downstream clustering algorithm. We use UMAP (Uniform Manifold Approximation and Projection) [13] for non-linear dimension reduction[4]. UMAP has three key hyperparameters: the number of components, number of neighbors, and a distance metric. The number of components determines the dimensionality of the reduction and we set it to 5. The number of neighbors balances local versus global structure; we set it to 15 to achieve a balance. For computing the distance between input vectors we choose cosine distance as it is a common choice for computing the correlation between text representations. After inputting $\mathbf{E}$ into UMAP, we are left with a reduced matrix $\mathbf{E_r}$ of size T x d_r where d_r is the reduced dimensionality corresponding to the number of components hyperparameter.

We then use HDBSCAN [12] to cluster the reduced embeddings. It is a density based method which allows it to be more robust to noise and outliers. It does not force each of the input points to be in a cluster, instead it can choose to classify points into a noise cluster. There are three parameters that can significantly impact the resulting clusters; minimum cluster size, minimum samples, and cluster selection epsilon. Minimum cluster size determines the minimum number of points that must be in a cluster. Minimum samples intuitively provides a measure of how conservative the clustering should be, with larger values leading to more points classified as noise. Cluster selection epsilon is used to merge clusters and is exactly the distance between clusters that will be combined. For example, if this parameter is set to 0.5, clusters within 0.5 units will **not** be separated. We found that both minimum cluster size and minimum samples have significant impact on the resulting clusters and are dataset dependent[5]. Cluster selection epsilon is useful in reducing the size of clusters since HDBSCAN does not have a number of cluster parameter, but there is a very fine threshold between merging a few clusters and merging them all into just two or three. The result of this step is a mapping of each of the T tweets to a cluster $c \in C$.

Once each tweet is assigned a cluster, it is still a challenge to determine how to get interpretable and presentable cluster names. We follow the approach from [5] and for each cluster c, concatenate all associated tweets into **one**

[3] Sentence-Transformers pretrained models

[4] An intuitive explanation of UMAP works can be found at umap-learn.readthedocs.io

[5] Further discussion of HDBSCAN parameters can be found at hdbscan.readthedocs.io

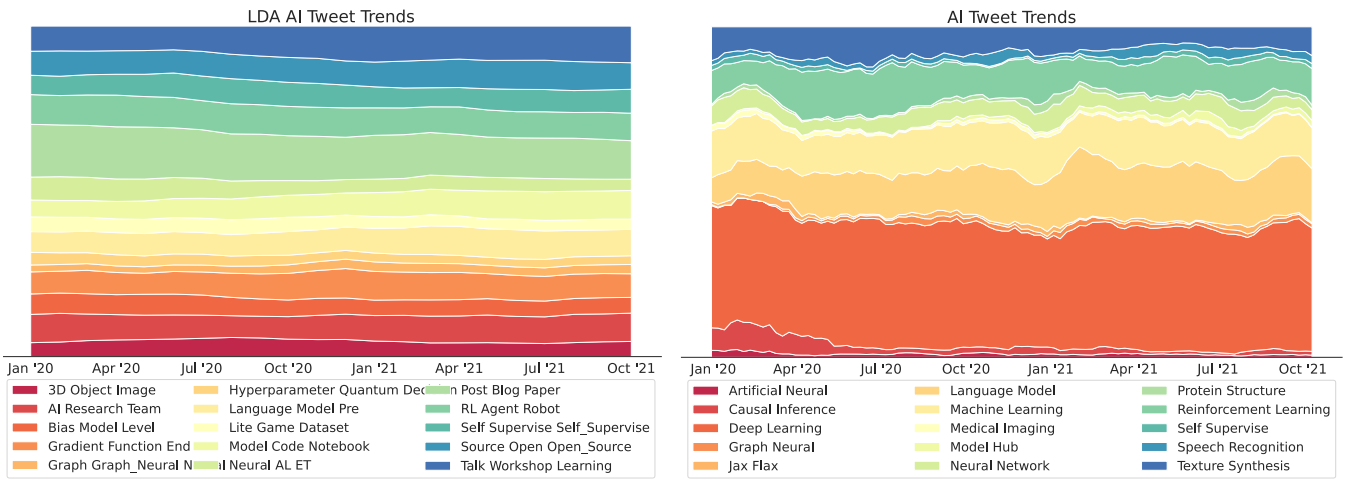

Figure 2: A comparison of the topics over time for LDA (left) versus our method (right).

document. Over this new set of documents, C_g, having size the number of clusters $|C|$, we convert the documents into a matrix of **bigram** token counts C_g x V where V is the vocabulary of bigrams. The use of only bigrams here was critical to giving us coherent topics. We hypothesize this is a result of the phenomena that many common topics in AI consist of two words (further analysis in results section). Also, this process is heavily dependent on the topic counts, so we apply the same corpus independent transformations for each tweet as we did for LDA. Using the bigram counts, we compute TF-IDF scores for each bigram in each document as follows:

$$TFIDF(t) = tf(t,d) * idf(d) \tag{1}$$

$$idf(t) = log(|C_g|/df(t)) + 1 \tag{2}$$

where t is a term $\in V$, d is a document $\in C_g$, tf is a count of t in d, df is the frequency of t in all documents. The result is a C_g x V matrix S which has the TF-IDF scores for each term in each document.

Using the TD-IDF scores for each term within each document, the extracted name of each cluster becomes the term (bigram in our case) with the highest score. To make the final cluster outputs more human-friendly, we capitalize the first letter of each token, fully capitalize tokens with 2 or less letters, and for subtopics ignore bigrams that have the same name as the overall topic (e.g. we found that a subtopic of "Reinforcement Learning" might also be "Reinforcement Learning" which is undesirable).

4. Experimental Setup

We start by using the entire processed dataset as input to LDA and *Embed, Cluster, Extract*. Both methods have a large amount of hyperparameters that can dramatically affect the final topics. For example, a key hyperparameter in LDA is choosing the number of topics. To this end we ran our model pipeline trying different hyperparameters and evaluating them using the charts shown in Figure 2. For LDA to extract the topic text, we take top 3 n-grams with the highest probability. Generally more than 3 words would lead to topics that are too long and include many common or extraneous words. For our method we use the methodology described in the previous section. The most critical hyperparameters to select were HDBSCAN's minimum cluster size, minimum samples, and cluster selection epsilon. Generally, we tried to force a relatively small number of topics because we have the secondary objective of creating a hierarchy of topics by again applying topic modeling to the original topics. We settled on minimum cluster size of 200, minimum samples of 150, and a cluster selection epsilon of 0. We found these parameters achieved the best balance of creating large enough high level topics to have enough samples to preform topic modeling again on the subset of data corresponding to each topic cluster. As a note, we chose to use this hierarchical structure rather than setting the parameters to create many topics from the beginning in order to achieve better interpretability for an end user (i.e. viewing 100 topics at once would be overwhelming, doing it this ways gives us a more interpretable hierarchy). A discussion of setting these hyperparameters for subtopics can be found in Appendix A.1.

Both methods are first applied on the entire tweet dataset, then re-applied to each topic to get the hierarchy of subtopics. After applying the topic modeling method on the dataset, we have a mapping from tweets to their topic and subtopic. To create the area charts, we aggregate the number of tweets belonging to a particular topic by week. Each line in the figure corresponds to the 7 week rolling av-

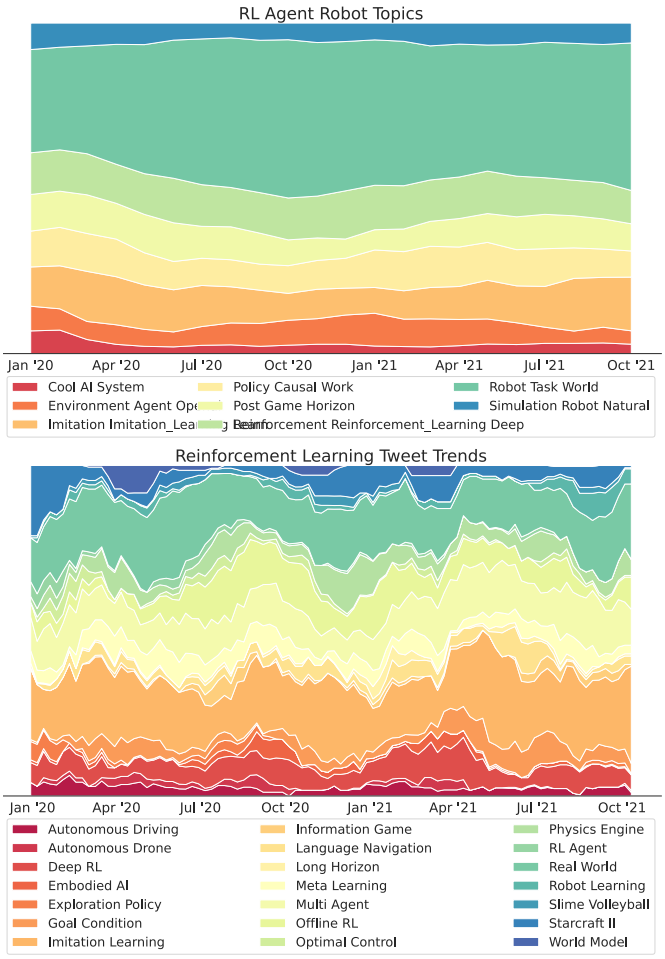

Figure 3: A comparison between subtopics generated for a reinforcement learning topic over time for LDA (top) and our method (bottom).

erage of tweet counts for that topic in order to smooth the plots. The data is normalized for the total volume of tweets for each time period. An example of how to intrepet the chart is as follows: Looking at the chart on the right in Figure 2, the area corresponding to "Deep Learning" can be interpreted as about one third of all tweets during any time period shown are related to this topic.

5. Results

Quantitatively evaluating the success of this project is difficult due to its unsupervised nature. Previous works like [8] and [21] use crowdsourcing for human evaluation or are able to compute metrics like precision due to the semi-supervised nature of their evaluation datasets. Neither of these are feasible for this project, so we focus on qualitative analysis leveraging our own domain knowledge of the problem.

Based on our experience with this dataset, we have topics in mind that we would like the model to generate (e.g. natural language processing, datasets, tutorials, and reinforcement learning). Furthermore, an example for expected subtopics within reinforcement learning might be; "multi-agent", "human in the loop", "environments", and "offline RL". Given our knowledge of the dataset, the topics generated by LDA in Figure 2 are not exactly what we want. We see some good topics such as "... Open_Source", "RL Agent Robot", and "Language Model Pre". However many topics are not cohesive and specific enough to be considered as subfields within AI. For example, there are many generic topics such as "AI Research Team" or "Neural AL ET". Many topics generated are around the type of content that the tweet contains like "talk", "papers", or "posts" rather than the actual topics of those tweets. Additionally some topics like "Bias Model Level" contain extraneous words which also makes it more difficult to determine a consistent number of top words to use for the topic's text. The relative trend of each topic seems to be quite similar over time, which is not what we would want because it makes it hard to discern trends. In the same figure, the chart on the right is our method and exhibits trends and topics that are more in line with our expectations for an optimal system. For example, the clear largest four topics are (in descending order): "Deep Learning", "Language Model", "Machine Learning", and "Reinforcement Learning" which correspond to our domain knowledge that these are some of the largest and topics in AI (although machine learning is a too generic of a topic and we would prefer not to see it). Similarly, the clear smallest topics are ones like "Graph Neural [Networks]" and "Speech Recognition", which are still important and large topics, but definitely not as common as the top four. Figure 3 shows a comparison between reinforcement learning related subtopics for each method. The results are similar to the primary topics. The LDA method generated fairly good topics, but still suffers from a lack of cohesiveness and a distinction between the amount of tweets in each category. We hypothesize that this is a result of LDA not having to freedom to assign samples to a noise category and being forced to generate n number of topics. HDBSCAN has the freedom to not assign samples to any category and create a new topic if there are enough. We see the power of this in the reinforcement learning trends as it came up with 21 subtopics, with most being very relevant (perhaps with the exception of "Information Game"). Additionally, we see the trends over time being fairly informative about the relative importance of each topic.

Table 1 shows all the full tweet texts for two subtopics, Starcraft II and Video Completion, during the month of September 2020. We chose the Starcraft II subcategory because it shows the power of the sentence transformer embeddings, but also highlights the weakness of our topic

Tweet Full Text	Topic / Subtopic
Don't miss today's panel discussion with Vladimir Kramnik, @DanielRensch and @Deepmind researchers on #AlphaZero and the new chess variants, in 3.5 hours from now!	Reinforcement Learning / Starcraft II
Our first tweet and our first blog! Live debugging production @golang applications using eBPF. Our CEO/Co-founder @zainasgar shares how to use gobpf and uprobes to build a function argument tracer for Go:	Reinforcement Learning / Starcraft II
In a bid to explore new frontiers in chess, our researchers worked with Vladimir Kramnik to use AlphaZero to test nine new variants of chess. The result? A more creative and collaborative relationship between chess players and machines.	Reinforcement Learning / Starcraft II
Section 4.1 of "A Machine Learning Perspective on Predictive Coding with PAQ" discusses old streaming language models to play games by opponent move prediction. It would be nice to see apps like this one using the latest neural models.	Reinforcement Learning / Starcraft II
Impressive work. Motion capture of my dreams https://t.co/mzj7ibTcs0	Texture Synthesis / Video Completion
Make people "disappear" with Flow-edge Guided Video Completion	Texture Synthesis / Video Completion
Super excited to share our work on video completion at #ECCV2020! Our method seamlessly removes objects, watermarks, or expands field-of-view from casually captured videos. Paper:	Texture Synthesis / Video Completion

Table 1: Full text of tweets from two different subtopics found using our method during the time period September 1, 2020 to October 1, 2020.

name selection method. None of the tweets are actually about Starcraft II, but they each discuss games under the context of reinforcement learning which tells us that original embeddings for these tweets must have been close to other tweets that mentioned Starcraft II. However, this raises the concern that Starcraft II is not a great title for this subtopic and likely was only chosen by our TF-IDF based method simply because it was the bigram that occurred most frequently. We also chose to highlight the "Video Completion" subtopic because of its quality in that each tweet is directly related to the subtopic and the subtopic is related to the topic of "Texture Synthesis". Examples like this showcase the power of our system to discover interesting subtopics and then get further information about the topic via the corresponding tweets.

6. Limitations and Future Work

Here we discuss the limitations and challenges of our method while also providing some ideas for future improvement. One major difficulty arises in tuning the hyperparameters of HDBSCAN. We found it very common to be in a case where we either have too many topics (30+) or too little (2-3). This phenonoma of too little topics can be seen in the Appendix Figures 4, 9, and 10. Finding a set of values for minimum cluster size, minimum samples, and cluster selection epsilon that generates a satisfying amount of topics for every subtopic is near impossible and seems like it would require manual tuning for each case. This poses an issue for deployment in a real world scenario where new samples come in constantly and the distribution of topics would change over time. An adhoc approach to this would be use heuristics to tune the hyperparameters automatically. For example, if we have too few topics we could gradually decrease the minimum cluster size and cluster epsilon until we have over a set threshold amount of topics. Perhaps a more robust solution would be to try other clustering algorithms that maintains some of the flexibility of HDBSCAN, while also us to select a number of clusters such as Agglomerative Clustering [14] or BIRCH [22]. We could also remove the need for a clustering algorithm altogether by using an autoencoder based method.

Another challenge is extracting coherent topic names. In our work we effectively leverage a hack where we use the phenomenon that bigrams in our dataset often make for great topic names. However, by using bigrams we are assuredly creating suboptimal names by missing out on other combinations of n-grams. The embedding based method from Top2Vec [1] where they use the topic centroid vector from HDBSCAN to find the most similar word vector may be worthwhile to try. However, it will likely still have the issue of coherence as there still is no relationship between the word vectors. A better approach might be to explore text generation or summarization models like GPT-2 [16] or BART [10], additionally pre-trained on our Twitter dataset, to generate text given an input embedding with the hopes that the generated text would could provide a short summary of the topic.

Our document embeddings sit at the bottom of the stack which makes it difficult to understand its downstream impact on the rest of system. A worthwhile test would be to try a lower resource embedding like GloVe or a sentence transformer with fewer parameters. If our topic quality degrades, we could hypothesize that the embeddings are in fact signif-

icant, so therefore a more custom method or larger model to create embeddings may lead to better results. In this case we could try newer sentence transformers like Phrase-BERT or try continuing pre-training our MPNet based model with our Twitter dataset.

7. Conclusion

In this project we created a topic modeling based system to help achieve our goal of distilling the breath of AI knowledge that is shared on Twitter. We explored language model Transformer methods of embedding sentences, algorithms for dimensionality reduction, algorithms for clustering, and algorithms for extracting topics. We showed qualitatively that the Embed, Cluster, Extract paradigm generates topics that are noticeably more coherent than those produced by LDA. Additionally, we showed that the our method is much better at producing trends over time than LDA. In future work we will look towards solving some of the issues discussed that make it difficult to deploy this system at scale (namely around the manual tuning that is currently required).

References

[1] Dimo Angelov. Top2vec: Distributed representations of topics. *arXiv preprint arXiv:2008.09470*, 2020. 2, 6

[2] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003. 2, 3

[3] Ricardo JGB Campello, Davoud Moulavi, and Jörg Sander. Density-based clustering based on hierarchical density estimates. In *Pacific-Asia conference on knowledge discovery and data mining*, pages 160–172. Springer, 2013. 2

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. 2, 3

[5] Maarten Grootendorst. Bertopic: Leveraging bert and c-tf-idf to create easily interpretable topics., 2020. 1, 2, 3

[6] Matthew Hoffman, Francis Bach, and David Blei. Online learning for latent dirichlet allocation. *advances in neural information processing systems*, 23:856–864, 2010. 3

[7] M Honnibal, I Montani, and S Van Landeghem. Boyd. *A. spaCy: industrial-strength natural language processing in Python*, 10, 2020. 3

[8] Mohit Iyyer, Anupam Guha, Snigdha Chaturvedi, Jordan Boyd-Graber, and Hal Daumé III. Feuding families and former friends: Unsupervised learning for dynamic fictional relationships. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1534–1544, 2016. 3, 5

[9] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *International conference on machine learning*, pages 1188–1196. PMLR, 2014. 2

[10] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*, 2019. 6

[11] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019. 2

[12] Leland McInnes, John Healy, and Steve Astels. hdbscan: Hierarchical density based clustering. *Journal of Open Source Software*, 2(11):205, 2017. 3

[13] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018. 2, 3

[14] Daniel Müllner. Modern hierarchical, agglomerative clustering algorithms. *arXiv preprint arXiv:1109.2378*, 2011. 6

[15] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014. 2

[16] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019. 6

[17] Juan Ramos et al. Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*, volume 242, pages 29–48. Citeseer, 2003. 3

[18] Radim Řehůřek and Petr Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May 2010. ELRA. 3

[19] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019. 2, 3

[20] Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. Mpnet: Masked and permuted pre-training for language understanding. *arXiv preprint arXiv:2004.09297*, 2020. 3

[21] Shufan Wang, Laure Thompson, and Mohit Iyyer. Phrase-bert: Improved phrase embeddings from bert with an application to corpus exploration. *arXiv preprint arXiv:2109.06304*, 2021. 3, 5

[22] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. Birch: an efficient data clustering method for very large databases. *ACM sigmod record*, 25(2):103–114, 1996. 6

A. Appendix

A.1. All Discovered Subtopics

Figures 4 - 10 show the subtopic trends for the remaining 14 subtopics that were generated by our topic model shown in Figure 2. Note that the same hyperparameters

were applied for each subtopic. For HDBSCAN we used a minimum cluster size of 25, minimum samples of 2, and a cluster selection epsilon of 0.24. Certain subtopics such as "Causal Inference" or "Self Supervise" that were small have very few subtopics which tells us that the cluster selection epsilon was set to high and many clusters were merged, or the minimum cluster size was set too high and few clusters could be generated. Setting the hyperparameters in this way was a tradeoff for large topics such as "Language Model" which tended to have very large amounts of clusters without a high cluster selection epsilon.

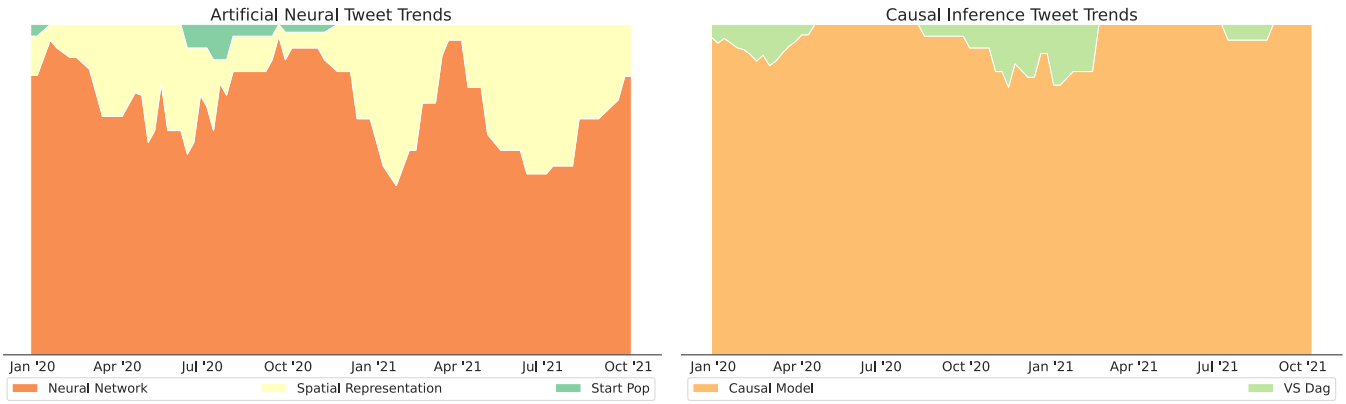

Figure 4

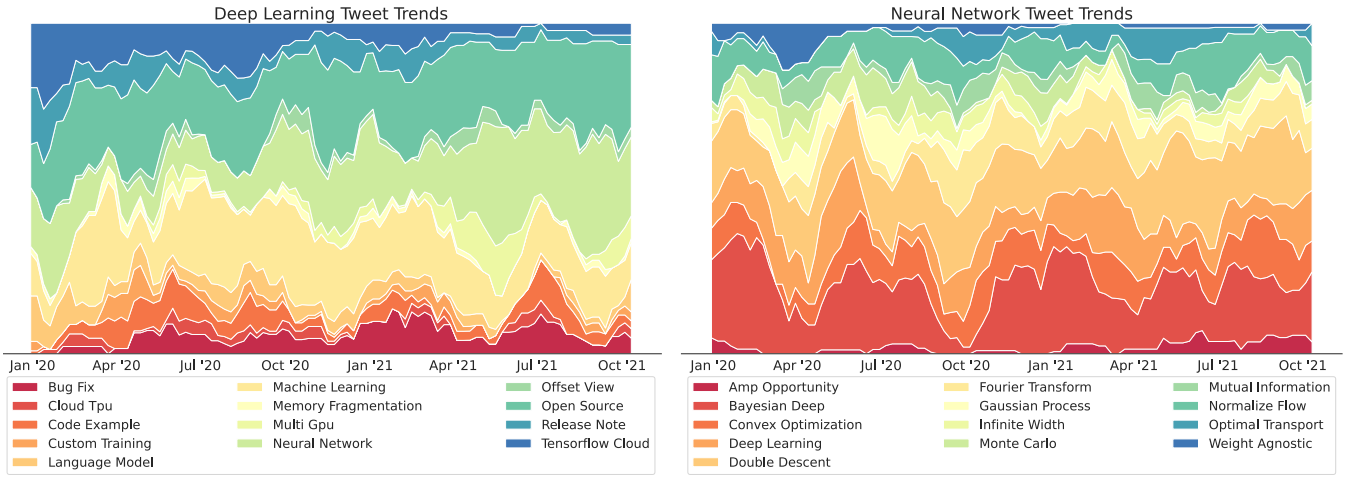

Figure 5

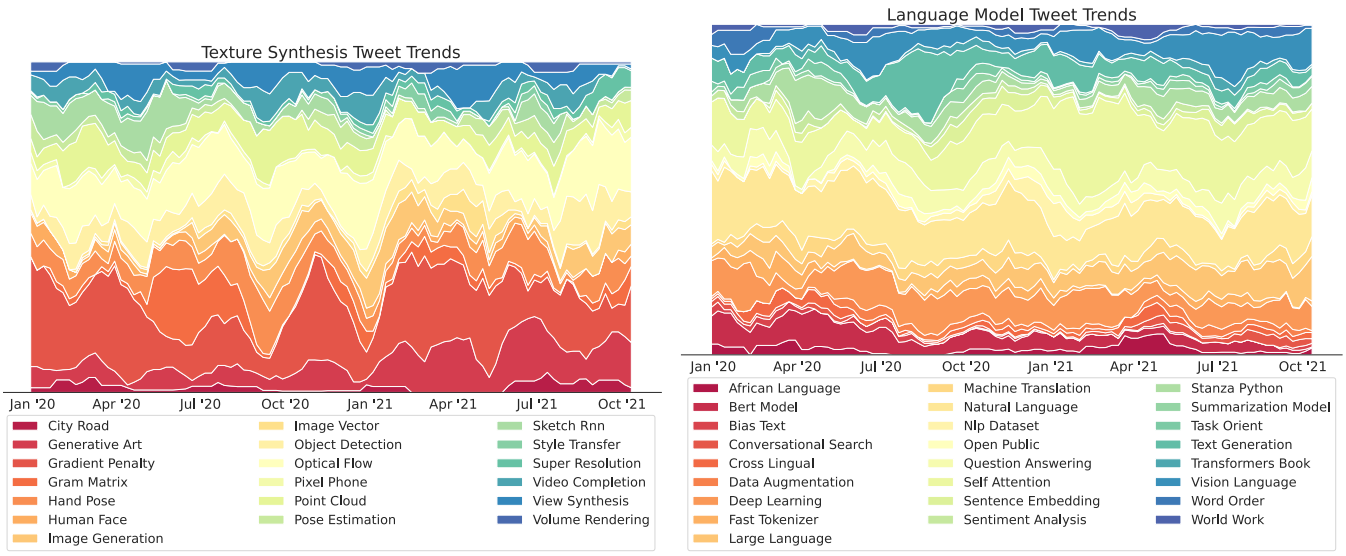

Figure 6

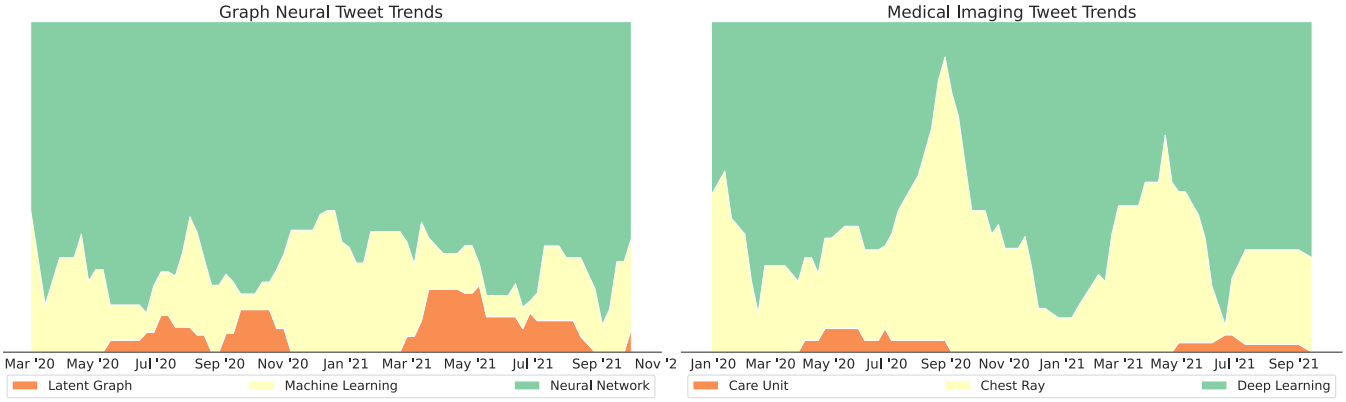

Figure 7

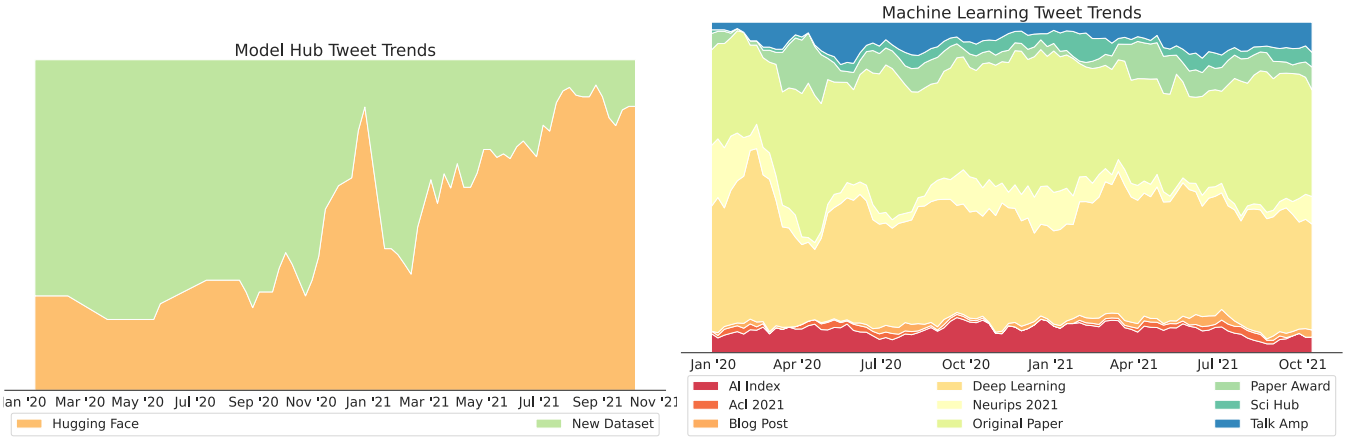

Figure 8

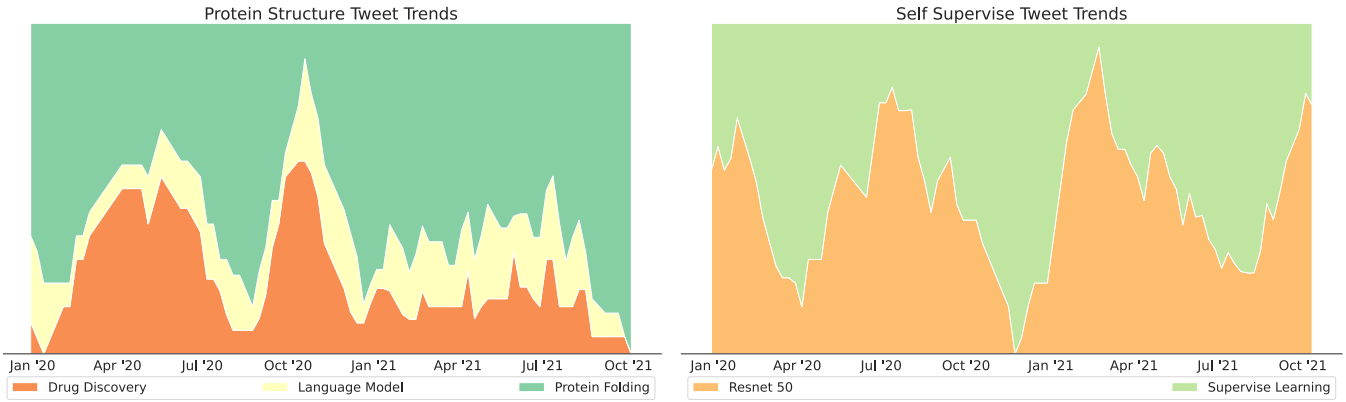

Figure 9

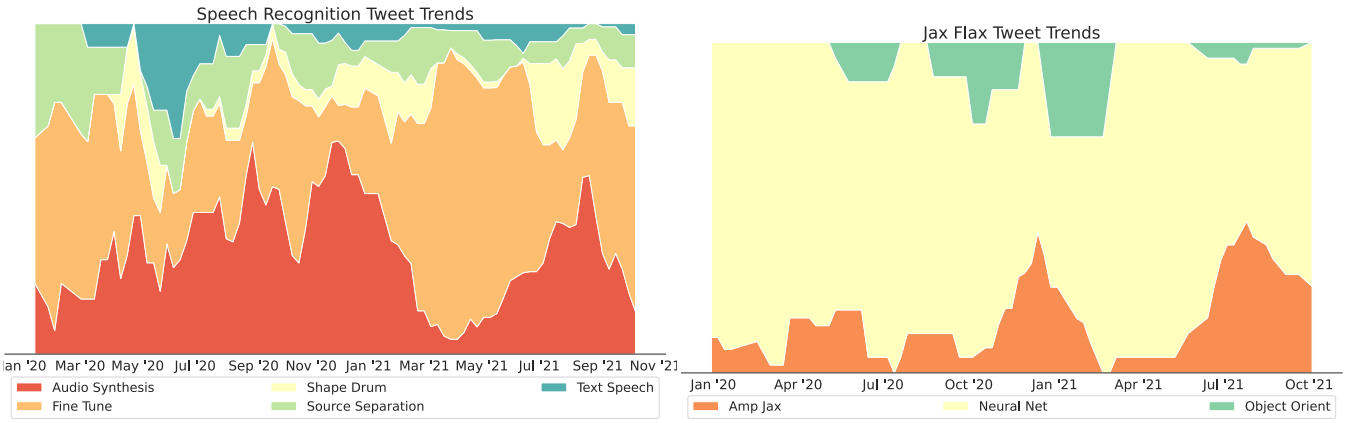
Speech Recognition Tweet Trends
ian '20
Mar '20
May '20
Jul '20
Sep '20
Nov '20
Jan '21
Mar '21
May '21
Jul '21
Sep '21
Nov '21
Audio Synthesis
Fine Tune
Shape Drum
Source Separation
Text Speech
Jax Flax Tweet Trends
Jan '20
Apr '20
Jul '20
Oct '20
Jan '21
Apr '21
Jul '21
Oct '21
Amp Jax
Neural Net
Object Orient

Figure 10