EasyTopics - NLP Systems Final Project

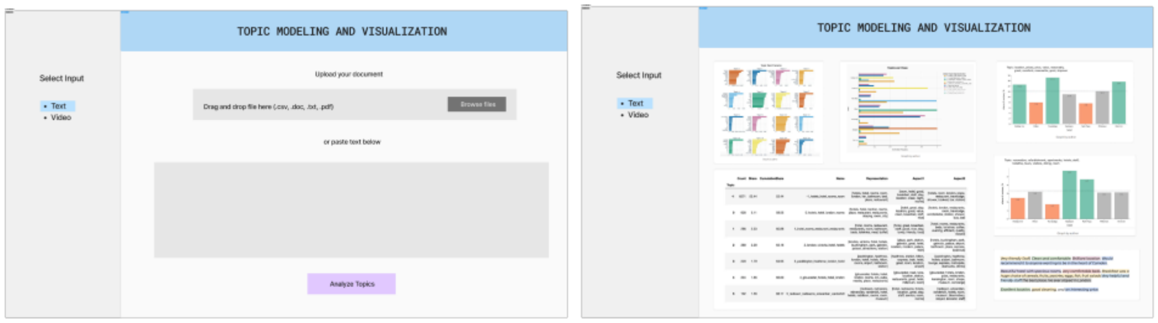
Chris Tam, Ian Bulovic, Nhi Le

We present EasyTopics, a web-based application designed to make topic modeling maximally accessible to non-developers. Our motivation is to abstract technical complexities such as file preprocessing or model parameter tuning away from our users so that they can more quickly and easily identify clusters and patterns in their data. EasyTopics can be accessed right now, either on the Internet through Streamlit Cloud or by running a Docker image locally. See the README.md in the repository for download instructions.

Topic modeling is an NLP task that is useful for clustering and labeling a set of documents in an unsupervised manner. Historically, this has been done with statistical approaches such as LDA, MNF, and LSA that aim to infer topics directly based on word frequencies. However, the most recent developments in this area, e.g. BERTopic and Top2Vec, instead involve calculating embeddings for documents using pretrained language models and then performing clustering on those embeddings.

For this project, we focus on BERTopic, wrapping its functionality into a web application. EasyTopics offers a streamlined interface that allows users to upload their own custom documents and document datasets, train topic models, and visualize BERTopic's results. Along with Python, we leverage several development tools that we've learned in this class, such as Streamlit, SQLAlchemy, and Docker, to make this possible.

Implementation and Roadmap


TOPIC MODELING AND VISUALIZATION
Select Input
Text
Video
Upload your document
Drag and drop file here (.csv, .doc, .txt, .pdf)
Browse files
or paste text below
Analyze Topics
TOPIC MODELING AND VISUALIZATION
Select Input
Text
Video

Early mockups

We began developing EasyTopics by outlining webpages and user flow with Figma mockups. We envisioned a file upload page supporting multiple formats, from text to pdf and video, as well as a results page displaying different visualizations.

Using these designs, we began implementing three basic webpages using Streamlit:

- File Upload - where users upload documents from their filesystem to the app
- Document View - where users can view and organize uploaded docs
- Topic Modeling Visualizations - where BERTopic is run and results displayed

We utilize special Streamlit features such as session state, progress bars, and spinners to enhance the user experience on each of these pages, and the specific implementation for each of them is described below. Throughout our development process, we employed conda environments to ensure our codebase worked across our different setups, and tracked and merged changes through separate GitHub branches and pull requests.

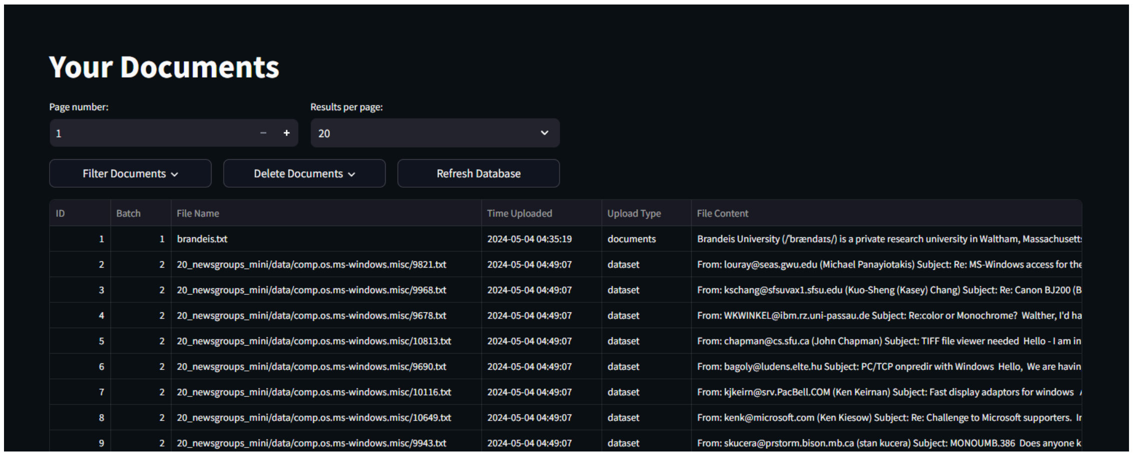
File Upload

We decided to split the functionality of our app into two distinct, but interacting use cases. The first use case is single document uploads, which can be run through an existing BERTopic model individually to receive predictions. The second is batch document uploads (ideally >200 documents), which can themselves be used to fit a new BERTopic model.

Whenever a file is uploaded, whether it be a txt or docx, it is converted from a BytesIO stream to a normalized UploadedDocument object with a content and filename field. All UploadedDocuments are then saved to our internal database (explained in the "Document View" section).

For batch uploads, our app expects documents to be compressed into a .zip file upon upload. Upon submission, our app uses the zipfile library to extract the files to a temporary directory and saves each file to the database separately as part of a single batch.

Document View


Your Documents

Page number: 1

Results per page: 20

ID	Batch	File Name	Time Uploaded	Upload Type	File Content
1	1	brandeis.txt	2024-05-04 04:35:19	documents	Brandeis University (/ˈbrændaɪs/) is a private research university in Waltham, Massachusett
2	2	20_newsgroups_mini/data/comp.os.ms-windows.misc/9821.txt	2024-05-04 04:49:07	dataset	From: louray@seas.gwu.edu (Michael Panayiotakis) Subject: Re: MS-Windows access for the
3	2	20_newsgroups_mini/data/comp.os.ms-windows.misc/9968.txt	2024-05-04 04:49:07	dataset	From: kschang@sfsuvax1.sfsu.edu (Kuo-Sheng (Kasey) Chang) Subject: Re: Canon BJ200 (B
4	2	20_newsgroups_mini/data/comp.os.ms-windows.misc/9678.txt	2024-05-04 04:49:07	dataset	From: WKWINKEL@ibm.rz.uni-passau.de Subject: Re:color or Monochrome? Walther, I'd ha
5	2	20_newsgroups_mini/data/comp.os.ms-windows.misc/10813.txt	2024-05-04 04:49:07	dataset	From: chapman@cs.sfu.ca (John Chapman) Subject: TIFF file viewer needed Hello - I am in
6	2	20_newsgroups_mini/data/comp.os.ms-windows.misc/9690.txt	2024-05-04 04:49:07	dataset	From: bagoly@ludens.elte.hu Subject: PC/TCP onpredir with Windows Hello, We are havin
7	2	20_newsgroups_mini/data/comp.os.ms-windows.misc/10116.txt	2024-05-04 04:49:07	dataset	From: kjkeirn@srv.PacBell.COM (Ken Keirnan) Subject: Fast display adaptors for windows
8	2	20_newsgroups_mini/data/comp.os.ms-windows.misc/10649.txt	2024-05-04 04:49:07	dataset	From: kenk@microsoft.com (Ken Kiesow) Subject: Re: Challenge to Microsoft supporters. Ir
9	2	20_newsgroups_mini/data/comp.os.ms-windows.misc/9943.txt	2024-05-04 04:49:07	dataset	From: skucera@prstorm.bison.mb.ca (stan kucera) Subject: MONOUMB.386 Does anyone k

Your Documents page - database viewable as interactive table

The “Your Documents” page allows users to manage any documents that have been uploaded to the internal database through a simple interface. Documents are displayed in an organized table, and can be filtered by their attributes or selectively deleted.

Database operations such as the file saving above and the features showcased on this page are handled by the operations.py module, which uses SQLAlchemy as an Object-Relational Mapper (ORM) to an SQLite database. The module supports selecting or deleting either individual files or a batch of them - all this functionality is covered with unit tests created using Python’s unittest package. Our current database schema is as follows:

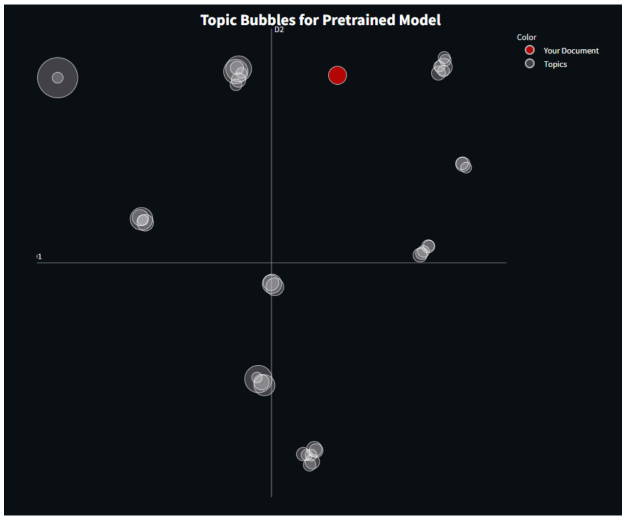
Documents

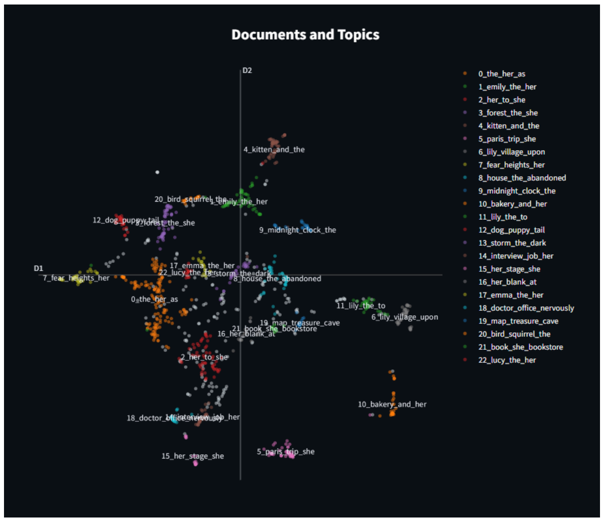
id	integer	unique identifier for the document
filename	string	location of the file on the user’s machine
batch_number	integer	identifier of the batch of documents this document is a part of
upload_time	datetime	time of document upload

content	text	text content of document
upload_type	string	‘document’ or ‘dataset’
topics	string	precomputed topic
probabilities	string	precomputed topic probability
model_names	string	name of the model created using this batch
path_to_models	string	filepath of the model created using this batch

Topic Modeling Visualizations

EasyTopics supports two modes of topic visualization, accessible with a batch selector at the top of the topic modeling page. The first is to use a pretrained model to classify a single new document with a topic and probability, and visualize that document in relation to the other documents it was trained on (left, below). The second is to visualize a trained model as a whole, using BERTopic’s built-in visualizations to, for example, display all documents in an embedding space (right, below).





The first visualization is our custom extension of BERTopic’s intertopic distance map that additionally embeds the submitted document and displays it in red as part of the outputted Plotly chart. If the user has not uploaded a batch of documents yet, we allow them to use BERTopic Wikipedia, a pretrained model downloadable from HuggingFace, to perform targeted visualization.

For our model-based visualizations, we precalculate both reduced and full embeddings over the batch that created the model for speed (if applicable), and display a selection of BERTopic's many built-in visualizations, such as hierarchy and similarity graphs.

Containerization with Docker, unit tests, and deployment with Streamlit Community Cloud

Lastly, with strong focus on usability and reproducibility, we put some final touches to our project as follows:

1. Docker containerization: to make sure other people can easily recreate our app on their device, we included the option to build and run a Docker container app.
2. Unit testing: we included unit tests for our database operations such as saving documents and datasets to the database, getting or deleting documents from the database, etc.
3. Deployment: Streamlit offers the option to easily deploy and host apps through the Streamlit Community Cloud. Currently, our app does not have the option to authenticate users so it is not fully usable yet since technically speaking, everyone can see anyone's uploaded data. But if we include authentication in a future version of the app, redeployment should be straightforward.

Reflection

Developing EasyTopics was a rewarding experience that taught us how to create a fully functional and distributable NLP system from start to finish. As the main NLP component of our project involved wrapping BERTopic, we evaluate our project based on the ease-of-use and efficiency of its features. Here, EasyTopics meets our expectations - the user interface is intuitive and accessible, allowing users to easily run and use the app without requiring extensive NLP expertise, and the system processes large datasets and loads models reasonably efficiently.

Of course, there are many additional features we could not add to EasyTopics due to time constraints that we leave to future work. We could expand our file upload page to accept a wider variety of formats, even transcribable ones, such as .pdf, .mp3, .mp4, etc. Our database could also be expanded with additional complexity to handle multiple users, store document embeddings, and track user interaction and visualization history.

Finally, BERTopic has many interchangeable components that could be experimented with. For instance, the dimensionality reduction (UMAP) and clustering (HDBScan) algorithms have several tunable parameters that could improve performance. Also, different representation models can be used to give more human-understandable names to topics, such as KeyBertInspired and LLMs like ChatGPT.

Team contributions

The division of labor ended up quite close to our original plan. Chris worked on project scoping, topic modelling code, document visualization, and documentation. Ian worked on developing the main web pages through Streamlit, managing file uploads and inference, and implementing performance optimizations. Nhi worked on the database schema, additional visualizations, unit tests and containerization. This clear division of responsibilities allowed us to work efficiently and effectively.