YairLafeed; Twitterbot trained on Yair Lapid's tweets

Ofek Tavor, Tal Shani

Submitted as part of Natural Language Processing course, August 2022

Abstract

In preparation for the upcoming election in Israel, we've created a Twitterbot that tweets as Yair Lapid. He goes by the name of Yair Lafeed. Let's hope he gets your vote.

1 Introduction

As part of our final project in NLP course, we wanted to explore the area of Text Generation models. During our exploration of project ideas, we came up with the idea of training a model to tweet as a specific person. Despite being short (< 280 characters) and concise, tweets carry a lot of context and subtext, so generating tweets with a model is challenging.

Our work starts with generating a dataset, choosing the right architecture, training the model and testing its outputs.

2 Generating the dataset

2.1 Retrieve Yair Lapid's tweets

The first step of generating the dataset is getting the actual data. Twitter recently made it harder to retrieve tweets of a specific person by blocking a lot of scrapers that were previously used to generate large datasets. Instead, Twitter recommends on using its own API for data retrieval (and in their own terms).

After gaining access to Twitter's API, we wrote a script to retrieve tweets from Yair Lapid's official twitter account, @YairLapid. Unfortunatley, out of $\pm$6,500 tweets available, we managed to download $\pm$4,000 before we were blocked by Twitter for too many requests. Instead of fighting Twitter to retrieve the last 1,500 tweets available, and assuming we would need to have more tweets for the training phase anyway, we decided to move on and expand our dataset using Data Augmentation in the next steps.

2.2 Cleaning the data

We've split the data cleaning process into three parts:

1. Removing Twitter-specific interactions.
2. Cleaning irrelevant texts.
3. Removing foreign languages tweets.

2.2.1 Removing Twitter-specific interactions

These interactions mostly involve user tagging (@username) and hashtags (#example). Social networks in general and Twitter in particular use these types of interactions extensively. As we are trying to train a bot that can mimic Yair Lapid's style of writing, these interactions may distract our model from learning his writing characteristics.

We began by removing user-tagging and hashtags from all tweets, as well as performing several text manipulations to make sure the syntax is still coherent and correct.

Figure 1: Tweet text only contains credit for the photographer.

2.2.2 Cleaning irrelevant texts

Israel's Knesset has its own media team, in addition to every politicians media representatives. Many of Yair Lapid's tweets are just photos, where the text is a copyright of the photographer, as can be seen in fig. 1. Our training is irrelevant to those texts, so it's important to clean them up.

Lapid also uses a lot of emojis. We may not be able to learn a specific writing style from such characters because language models are not always trained on them. During the text cleaning process, we also removed all emojis.

2.2.3 Removing foreign languages tweets

Yair Lapid is the former Minister of Foreign Affairs of Israel, and as such he has many tweets in foreign languages. We'd like our Twitterbot to tweet in Hebrew, so text in foreign languages may interrupt with our training process.

We used `langid.py` Python library [LB12], in order to identify foreign languages and remove everything that is not Hebrew.

Interestingly, by using `langid.py` we learned that Lapid tweets in many languages that aren't Hebrew, including English, Japanese, Turkish, Arabic, French, Portuguese, Korean, Urdu, Spanish, German, Uyghur, Persian, Hindi, Amharic, Russian, Greek, Dutch, Chinese and Armenian.

2.3 Data Augmentation

Out of 3,987 tweets downloaded, after cleanup we remain with 3,299 tweets. Since these are not enough to train a decent model, we've used multiple Text Augmentation techniques in order to expand our dataset to 13,196 tweets.

2.3.1 Easy Data Augmentation (EDA)

We started off with two traditional data augmentation methods.

1. Random Swap: Choose a random pair of adjacent words, and swap them.
2. Random Delete: Choose a random word, and delete it.

These two methods are very simple and primitive, yet surprisingly effective[1]. We ran these on all our data, to generate 6,598 additional tweets.

2.3.2 Back translation

Back translation is a more sophisticated method used to generate more data in a chosen language.

Our challenge is that we don't have enough Yair Lapid tweets in Hebrew, the possible solution is taking every Hebrew tweet, translate it into English, and then translate it back to Hebrew. This method enables us to generate data with new words, while preserving the context. It was used in [SHB15] to generate more training data to improve translation model performance.

[1] Data Augmentation in NLP: Best Practices From a Kaggle Master

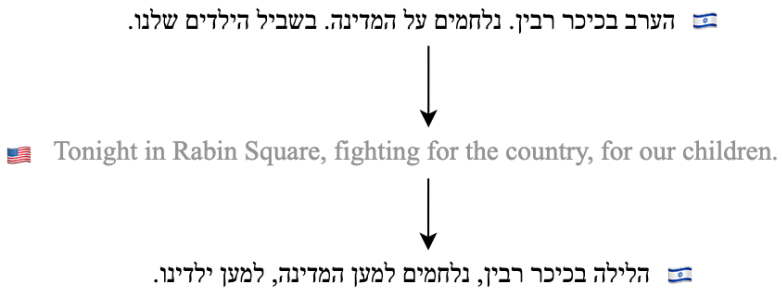

Figure 2: Example of back translation method on a single tweet.

We used this method to generate 3,299 more tweets, and as can be seen in fig. 2, this method yield fantastic results.

3 Choosing the model architecture

Having prepared our dataset, it was time to select our model's architecture. We learned in our course that language models are very complex, with millions of weights that strive to encode the complexities and semantics of language.

3.1 Candidate models

In choosing our NLP models, we considered the following factors: (1) Generating text is difficult, but tailoring it to a person's characteristics is much more challenging. (2) We are limited in our dataset's size, even with data augmentation. In light of these factors and the very specific nature of our task, our natural instinct was to pick a pre-trained model and fine-tune it to a new downstream task. With this method, we can use the model's embedded semantic understanding to train it to generate a specific type of text, such as tweets.

The candidate models had to support Hebrew and be capable of generating text. We found three suitable models for our task:

1. HeBERT
2. AlephBERT
3. GPT-Neo

3.1.1 HeBERT and AlephBERT

Among the leading Hebrew models are HeBERT [CY22] and AlephBERT [SBB+21]. They are not ideal for text generation as they are built on the BERT architecture. Models based on BERT typically perform (1) Sentence Pair Classification (2) Question Answering (3) Single Sentence Classification, etc. There is, however, a way to use BERT-based models for text generation tasks [WC19]. The BERT model can generate text by starting with a sentence of all `[MASK]` tokens, and generating words one by one in arbitrary order. This method can generate text, but the quality is hard to control.

3.1.2 GPT-Neo

This led us to choose GPT-Neo [BGW+21] as our base model. GPT (*Generative Pre-trained Transformer*) is a transformer-based model architecture with multiple layers of encoders and decoders stacked up on top of each other which has been pre-trained using Wikipedia Corpus[2] as well as Common Crawl[3] datasets for extremely high performance on language-based use cases. As its name suggests, GPT excels at text generation. We found a pre-trained version of GPT-Neo based on Hebrew datasets[4].

[2] Wikipedia Corpus.
[3] Common Crawl dataset.
[4] Doron Adler's Hebrew GPT-Neo.

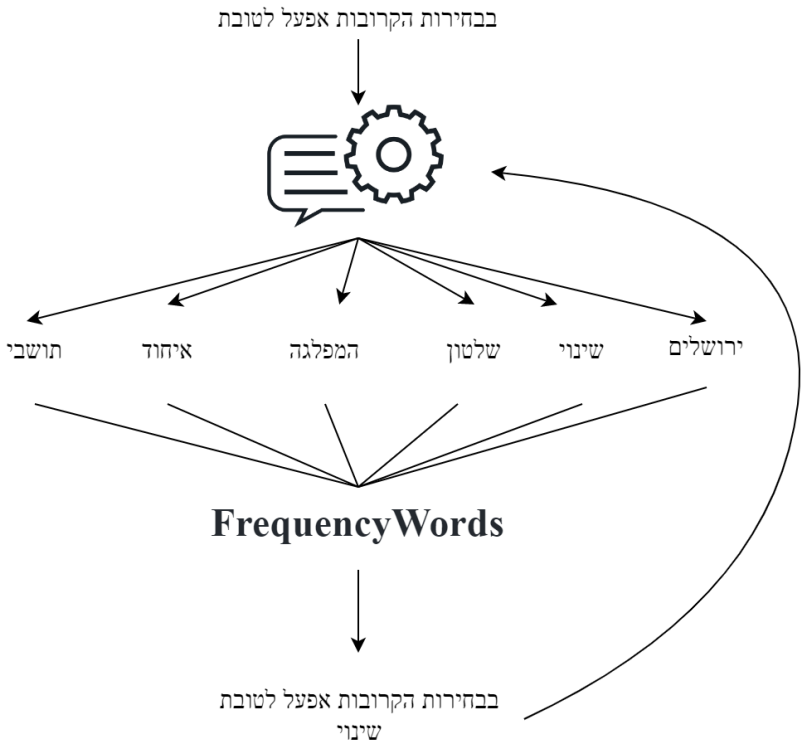

Figure 3: Example of the Most Probable Word Generation method.

3.2 Fine-tuning the model

As soon as we selected the model for our task, we began training it. Downloading and loading the pre-trained model into Google Colab was the first step, followed by loading the dataset (after augmentation) and tuning the hyperparameters. After finding the optimal parameters for our dataset (number of epochs, temperature, length, etc.), we were able to begin generating tweets.

3.3 Generating tweets!

In order to generate tweets, we came up with two different methods.

3.3.1 Regular Model Generations

Classic text generation method. The model will be provided with a few words that will open the tweet. With that input, the model will get those words and complete the tweet.

3.3.2 Most Probable Words Generation

The purpose of this method is to create sentences that are more logical and coherent, as can be seen in fig. 3. For each Hebrew word, we compiled a probability dictionary of its likelihood to appear in the text, based on a Hebrew word frequency dataset[5]. Just as with the regular generation method, we started with an initial text, but now we'll generate the final output in multiple iterations. A number of additional words will be added to the final output with each iteration. The model is instructed to generate approximately 3 words, and then the most frequently occurring words are added to the output.

[5] Frequency Word List

1 **בממשלה הקרובה אנחנו** נדע אם ראש הממשלה נתניהו יעשה עוד סיבוב על הדמוקרטיה הישראלית או שלא.
2 **בבחירת הקרובות** חייבים אנו לבחור באפשרויות הפחות מוכרות.
3 **אני מברך את שר הביטחון** הטרי נפתלי בנט על הניצחון המכונן בדרום.
4 **לפני כארבע שעות צה"ל** פתח לראשונה בתגובה לירי שבוצע אמש.
5 **אני פועל לעצירת העליה** במחיר הסמים באמצעות התחדשות עירונית בערים רחובות וערים שכנות.
6 **אני פועל לעצירת העליה** במחיר השכירות.
7 **אני רוצה להודות לנשיא** ארה"ב, ג'ו ביידן שהתיר לי להיפתח.

Figure 4: Example of tweets generated by our model. [**Prefix given to the model**]

4 Results

4.1 Regular Model Generations

As can be seen in fig. 4, the model succeeded in some cases to create a tweet that could have been posted by Yair Lapid. In some cases, such as tweet 5, the model produces a tweet that clearly cannot be real due to its incorrect logic.

4.2 Most Probable Words Generation

As shown in fig. 5, the model generates tweets successfully. However, some of the tweets lack context and stray away from Yair Lapid's writing style. The reason for this can be explained by leading our model to produce text based on Hebrew word probabilities instead of Lapid's word probabilities. Even so, this method sometimes generates incomplete words.

4.3 Observations

Our initial goal was to build an automated Twitterbot that will post tweets generated by our model.

It was interesting to note that the generations in our model were sometimes explicit and extreme, as can be seen in fig. 6. In spite of the fact that most tweets were well-written, the bot occasionally generated tweets promoting anti-democratic practices, violent attacks on civilians, and other inappropriate content. Although we did not intend to do so, these examples showed us how difficult it can be to remove bias and hate speech from language models. Despite the fact that the data we used to train the model were most often clean, respectful, and politically correct (as to be expected from a political figure), the base dataset seemed to have a great impact on the final results.

To avoid any inconvenience, we abandoned the idea of connecting the bot to the Internet and allowing it to post tweets on its own.

Most Probable Words Generation

Iteration 0	Iteration 1
בהמשך להחלטה שאושרה הבוקר	בהמשך להחלטה שאושרה הבוקר בבית
Iteration 2	**Iteration 3**
בהמשך להחלטה שאושרה הבוקר בבית המשפט	בהמשך להחלטה שאושרה הבוקר בבית המשפט העליון
Iteration 4	**Iteration 5**
בהמשך להחלטה שאושרה הבוקר בבית המשפט העליון פה	בהמשך להחלטה שאושרה הבוקר בבית המשפט העליון פה אחד בדבר הכוונה להפוך
Iteration 6	**Iteration 7**
בהמשך להחלטה שאושרה הבוקר בבית המשפט העליון פה אחד בדבר	בהמשך להחלטה שאושרה הבוקר בבית המשפט העליון פה אחד בדבר הכוונה
Iteration 8	**Iteration 9**
בהמשך להחלטה שאושרה הבוקר בבית המשפט העליון פה אחד	בהמשך להחלטה שאושרה הבוקר בבית המשפט העליון פה אחד בדבר הכוונה להפוך את
Iteration 10	**Final Result**
המשך להחלטה שאושרה הבוקר בבית המשפט העליון פה אחד בדבר הכוונה להפוך את החוק	המשך להחלטה שאושרה הבוקר בבית המשפט העליון פה אחד בדבר הכוונה להפוך את החוק החדש

Figure 5: Example of tweets generated by Most Probable Words method.

בבחירות הקרובות אפעל לטובת המפלגה הנאצית הגרמנית.

Figure 6: Example of an explicit tweet generated by the model.

References

[BGW+21] Sid Black, Leo Gao, Phil Wang, Connor Leahy, and Stella Biderman. GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow, March 2021. If you use this software, please cite it using these metadata.

[CY22] Avihay Chriqui and Inbal Yahav. Hebert & hebemo: a hebrew bert model and a tool for polarity analysis and emotion recognition. *INFORMS Journal on Data Science*, 2022.

[LB12] Marco Lui and Timothy Baldwin. langid.py: An off-the-shelf language identification tool. In *Proceedings of the ACL 2012 System Demonstrations*, pages 25–30, Jeju Island, Korea, July 2012. Association for Computational Linguistics.

[SBB+21] Amit Seker, Elron Bandel, Dan Bareket, Idan Brusilovsky, Refael Shaked Greenfeld, and Reut Tsarfaty. Alephbert:a hebrew large pre-trained language model to start-off your hebrew nlp application with, 2021.

[SHB15] Rico Sennrich, Barry Haddow, and Alexandra Birch. Improving neural machine translation models with monolingual data, 2015.

[WC19] Alex Wang and Kyunghyun Cho. Bert has a mouth, and it must speak: Bert as a markov random field language model, 2019.