

# Write-Up Robot Judge

Huber Simon

Department of Computer Science, ETH Zurich, Switzerland

*Abstract—*

## I. INTRODUCTION

Topic models are designed to group documents with similar content. A lot of this techniques are unsupervised for multiple reasons:

- 1) There is just not enough data that is already labeled.
- 2) The target data set is very small or very specific.

But if we interpret hashtags as labeled data for topics then there is already enough data labeled for a very wide range of topics on twitter. With 500 million tweets per day [1]. **Figure out how many percent are tagged with a topic..** The idea then was simple, learn a topic model **explain topic models** through hashtags and apply it to other domains. In this work we were interested in applying it to speeches of politicians. In particular we thought about the following ideas: How does the topics of a speech for a senator change if he is up for reelection ? How are the topics distributed for a democrat or for a republican senator ? Can we use hash tags to predict the party ? And lastly how do the topics compare between speech and twitter for a specific politician and to the general trendy topics ?

The paper continuous in the following manor: In section III we discuss the two models we used to predict the hash tags and why one failed. Then in section IV we described the planned and executed experiments as well as their result. In section V we discuss the results and finally in section VI we summarize and conclude the paper.

- General Idea - Hypothesis - Paper Structure

## II. DATA

In this section we present the used datasets, how they were obtained and how they looked.

### A. Twitter Data

Twitter has the rule that one is not allowed to save tweets along with text. This is because if one wants to make any changes to the tweets e.g. delete them etc. this is no longer globally possible if they are saved somewhere else. So one is only allowed to save them via an unique ID that can be used to retrieve the tweets with the twitter api. A databank full of this tweet ID datasets is found in [2]. We downloaded two datasets.

1) *Congress Dataset:* The 115th U.S. Congress Tweet Ids dataset contains 2,041,399 tweets. It covers a date distance from 2017-01-27 to 2019-01-02. It therefore covers 1 year before the senate elections and one year after the senate elections which makes it definitely very interesting for us.

2) *News Dataset:* The News Outlet Tweet Ids we also downloaded was a fallback dataset which in case the congress dataset was to "fuzzy" could be used to train on more well written, structured and diverse topic distributed dataset. Which was of course just an assumption.

### B. Speeches Data

The congressional records and therefore the speeches can all be publicly accessed on the congress page [3]. But this side makes it very hard to access specific information. So people developed software packages that parse this side for the information and presents it in an accessible format mostly json. I first tried a package from another student but due to quite a few bugs I had to switch to [4] which worked very nicely. I used it to parse all congress information from the years 2017 and 2018.

## III. PLAN

In the first part of this section I will tell shortly about the failure and massive time loss that resulted in trying to adapt LDA to be used with the twitter data and in the second part a short summary of the neural network based method that finally worked for predicting hash tags.

### A. Latent Dirichlet Allocation (LDA)

1) *Standard LDA:* LDA was one of the algorithms for topic models that got presented in class. The generative model to generate a word makes it very simple to think about. Given a document sample a topic and for this topic sample a word. And the word is the thing we observe and we can use to learn the corresponding distributions.

2) *Extended LDA:* So naturally I thought it should be simple to extend LDA so that it can also predict hash tags in the following way. For some document and its document-topic distribution I sample 2 topics and then for the topic-word distribution I sample a word and additionally to the topic-word distribution I just have an topic-hashtag distribution of which I sample a hashtag. And the we can learn the correct distributions since we observe both word and hashtag.

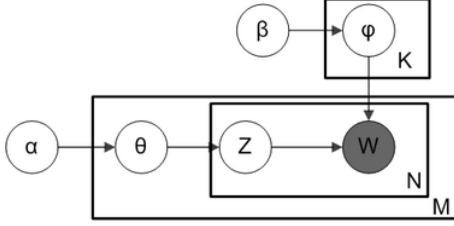


Figure 1. Plate notation for LDA with Dirichlet distributed topic-word distributions.

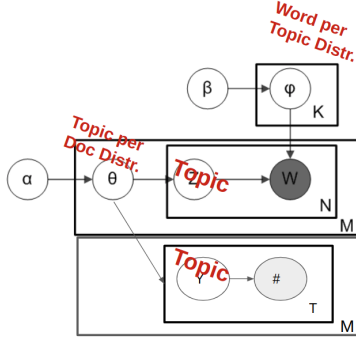


Figure 2. Plate diagram of extended version of LDA with additional distribution for hashtags for a given topic.

### B. Training

The plan ready I had to find some framework that allowed me to implement and train this generative model. I decided to use pyro [5] which allows to describe the generative model in a very pythonian way and uses pyTorch [6] as backend for matrix and optimization calculations which makes it ideal for big datasets. <https://github.com/SimiPro/robot-judge/blob/master/Untitled.ipynb> and cite as pyro implementation. The implementation can be seen in [cite my pyro impl.](#). But it just didn't work. I tried a multitude of different proposal distributions and even neural networks. It just didn't really converge nicely. I initially thought that the problem was my extension on top of LDA. But it didn't work for LDA aswell.

### C. Tweet2vec - Recurrent Neural Network

After the extended LDA somewhat failed I tried a multitude of other models. The one that finally worked was a neural network based solution: Tweet2Vec [?]. The basic idea behind this model is that they view the character as smallest unit rather than the word as in LDA. This brings quite a few advantages: **How many unique words are in the twitter dataset ? Which makes it difficult to use word based systems.**. Furthermore they assume that a tweet has some latent representation and from this latent representation they predict the hashtag. To get a loss they use cross entropy loss based on predicted hashtag versus given hashtag. The neural network is mainly built out of Bi-directional Gated

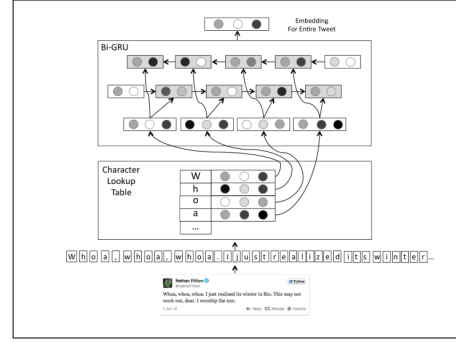


Figure 3. Tweet2vec embedding part of the neural network.

Recurrent Unit (Bi-GRU) which behave similarly as LSTM's (Long short-term memory) blocks. The size of the input is reasonable constraint on the number of characters a tweet can have - 145. The embedding part of the tweet2vec neural network can be seen in Figure 3. The decoder part is just a dense layer with the number of neurons equal to the number of hashtags. The output of this dense layer is then used as input for the softmax function to get the probability of each hashtag.

## IV. EXPERIMENTS & RESULTS

### V. DISCUSSION

### VI. SUMMARY

### REFERENCES

- [1] S. Aslam, "Twitter by the Numbers: Stats, Demographics," <https://www.omnicoreagency.com/twitter-statistics/>, 01.2019, [Online; accessed 07-July-2019].
- [2] "Tweet ID Datasets," <https://www.docnow.io/catalog/>, 01.2019, [Online; accessed 07-July-2019].
- [3] "Congress Page," [congress.gov/congressional-record](https://www.congress.gov/congressional-record), 01.2019, [Online; accessed 07-July-2019].
- [4] D. D. J. C. Judd, Nicholas and L. Young, "congressional-record: A parser for the Congressional," <https://github.com/unitedstates/congressional-record>, 2017, [Online; accessed 07-July-2019].
- [5] E. Bingham, J. P. Chen, M. Jankowiak, F. Obermeyer, N. Pradhan, T. Karaletsos, R. Singh, P. Szerlip, P. Horsfall, and N. D. Goodman, "Pyro: Deep Universal Probabilistic Programming," *Journal of Machine Learning Research*, 2018.
- [6] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in PyTorch," in *NIPS Autodiff Workshop*, 2017.