Predicting Stack Overflow Question Tags

Jan Viilma

School of Information Technologies
Tallinn University of Technology, Estonia
janviilma@hotmail.com

Abstract

The aim of this project was to predict the key words of a text in a larger text corpus. Specifically the Stack Overflow user questions dataset [1]. These key words are have been assigned by human users. Different options and their viabilities for solving this problem are discussed and compared. Only unsupervised models, such as latent Dirichlet allocation [2], are going to be covered in the experiments section below. It will be interesting to see how well these mathematical models can predict the key words that people have assigned.

1. Introduction

In machine learning and natural language processing (NLP) this kind of keyword prediction problem is usually referred to as topic modeling, which is an umbrella term for a variety of text mining and pattern recognition methodologies [3]. Text mining or data mining is essentially the construction of a statistical model that best describes this data [4]. Topic modeling can either use unsupervised probabilistic algorithms or supervised learning algorithms. One of the most popular unsupervised approaches is latent Dirichlet allocation. LDA also has a further developed analog, sLDA [5], which uses a supervised training approach.

2. Background

Topic modeling could be described as working through an article with a set of different colored highlighters. As you come upon a key word you highlight that word with one of the colors in such a manner that the key words in a similar theme have the same color. In practice this highlighting is determined by mathematics and a lot of different algorithms exist for that purpose.

2.1. Term frequency-inverse document frequency

Term Frequency times Inverse Document Frequency (TFIDF) is a measurement of how concentrated in a relatively few texts are the occurrences of a given word. The idea here is that if a text in a larger text corpus is about a certain topic, then there are a few infrequent words related to this discrete topic a lot more densely packed together than in other parts of the corpus [4]. It is usually calculated as follows,

$$TF_{ij} = \frac{f_{ij}}{\max_k f_{kj}} \quad (1)$$

where f_{ij} is the frequency of a term (word) i in text j and f_{kj} is the most often occurring term in the same text, meaning that $TF_{kj} = 1$. From this follows,

$$IDF_i = \log_2(N/n_i) \quad (2)$$

where N is the number of texts in the corpus and n_i is the number of texts that contain the term i. The score of a term i in text j is then $TF_{ij} \times IDF_i$. The terms with the highest scores in a text are now most likely going to represent this text. With this formula both the occurrences of a term in the whole corpus and every text individually are taken into account. This results in a term-by-text matrix X, where every element is the score of a single term in a single text, thus converting these texts into fixed-length lists.

2.2. Latent semantic analysis

In *latent semantic analysis* (LSA) or *latent semantic indexing* (LSI), a large matrix of term-text association data is used to create a "semantic" space wherein terms and texts that are more closely related are more close together [6]. The main problems that this method tackles are *synonymy* and *polysemy*. *Synonymy* means that multiple words can describe the same thing and *polysemy* refers to the fact that a single word can have multiple meanings. Mathematically this is done by for example using the TDIDF term-by-text matrix and applying singular value decomposition on it, which allows us to build a new matrix where terms and texts that are more similar are closer together in space (vectors that represent them have a smaller cosine).

The problem with this approach is that the algorithm doesn't differentiate well between terms and texts where the term could have multiple meanings, resulting in some texts being close to a set of terms that don't really describe those texts.

2.3. Latent Dirichlet allocation

Latent Dirichlet allocation is an unsupervised probabilistic topic model. The intuition of this model is that first we assume some number of topics for our text corpus. These topics are the distributions over all of the words. First we choose the distribution of chosen topics. Next we iterate over every word and see which topic it represents [7].

2.4. Hierarchical Dirichlet process

Another powerful topic model is *hierarchical Dirichlet process* (HDP) can be used [8]. It is a nonparametric mixed-membership model for unsupervised analysis of grouped data. This means that unlike LDA or LSA, HDP can infer the number of topics from the data.

2.5. Figures

All figures must be centered in the column (or page, if the figure spans both columns). Figure captions should follow each figure and have the format given in Fig. 1.

Figures should preferably be line drawings. If they contain gray levels or colors, they should be checked to print well on a high-quality non-color laser printer. If some figures contain bitmap images, please ensure that their resolution is high enough to preserve readability.

2.6. Tables

An example of a table is shown as Table 1. Somewhat different styles are allowed according to the type and purpose of the table. The caption text may be above or below the table.

Table 1: *This is an example of a table.*

ratio	decibels
1/1	0
2/1	≈ 6
3.16	10
10/1	20
1/10	-20
100/1	40
1000/1	60

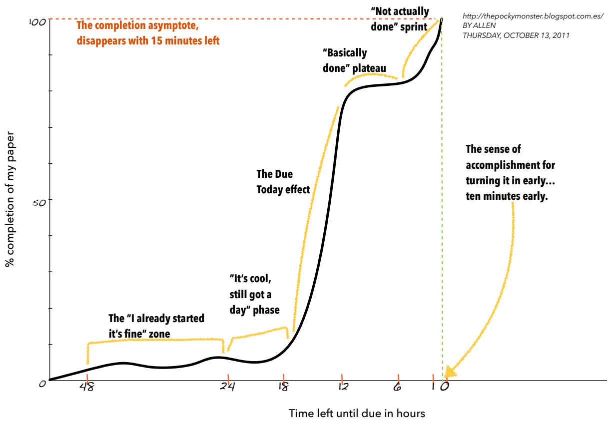

Figure 1: *This is an example of a figure.*

3. Approach

The approaches that are used in this project will be considering the properties and idiosyncrasies of the Stack Overflow dataset. Meaning that when a different type of dataset is used, another approach could give more precise results. The size of the used Stack Overflows tags dataset is 62.4 MB and consists of 3 750 994 entries with two values, the corresponding question id and value of the tag (Table 2),

Table 2: *Stack Overflows dataset of tags.*

Id	**Tag**
80	flex
80	actionscript-3
80	air
90	svn
...	...

When counting only the unique values, this count drops to 37 034. As we can see, this is a lot of keywords and a simple classification model training won't be enough. This is why we are going to use some of the mentioned unsupervised topic models and compare their results. This means that the tags are only going to be used in evaluation after training the model.

3.1. Preparing the data

When examining a few questions in the Stack Overflow dataset (Table 3) and comparing them to their corresponding tags, it can be seen that the "*Body*" or "*Title*" columns individually will not have enough information for creating an accurate model. All of the assigned tags will rarely be in a single column. In addition, all of the *topic models* that were discussed and even most that weren't mentioned above depend heavily on the density of rare words. This is why the input corpus is generated by merging the "*Title*" and "*Body*" columns.

Table 3: *Stack Overflows dataset of questions.*

Id	**Title**	**Body**
80	flex	asdd
80	actionscript-3	asd
...	...	...

Before starting any topic modeling the text corpus has to go through certain processes (assuming that the text corpus is a list of texts):

1. All of the texts have to be tokenized into lists of words.
2. It is recommended to filter out more often recurring "*stop words*" (*e.g.* "a", "the", "and") because they don't carry any real meaning about the topic. Other kinds of filtering methods could be used here. The end target here is to have as little unnecessary data as possible, considering the *garbage in, garbage* out principle [9].
3. A dictionary of the tokenized corpus has to be created where every word comes up only once and has a discrete index value.
4. The text that is used for model creation is now converted to *bag of words* representation, which means that every word is now represented by a vector of its index and number of occurrences.

Now the data is ready for further model training.

3.2. Model creation

The questions dataset size is 1.79 GB and it consists of 69 entries. It has some unnecessary columns in it, which we are going to ignore. The most important columns for us are the "*Title*" and the "*Body*". After throwing out all the unnecessary columns, the dataset is still quite large. This means that we have to be extra careful before doing any model training, because the training times are going to be long take up a lot of resources.

4. Experiments

All of the code was written in Python with *Jupyter Notebook*. They are made to be easily comprehensible and re-usable by making use of IPythons widget API. The application is divided into four main parts – data analysis, data preparation, model creation and model analysis. The main library for data loading and structure handling is done with *pandas* library. The most important library that all of the model building and data preparation is done with is *Gensim* [10]. Gensim is a Python library that has a wide variety of powerful tools for *topic modeling* and *document indexing* that can handle huge bodies of texts. It also has the tools for preparing your datasets for different deep learning algorithms if necessary. It has good community support and comprehensive well maintained documentation. *Gensim* also has wrappers for more popular external libraries, *e.g.* *keras*, *scikit-learn*.

4.1. Data preprocessing

Data preparation is partly done with *Gensim*'s built-in preprocessing methods. This will filter out all of the HTML tags, *stop words*, punctuations and redundant white-spaces. Next it will turn the text into lowercase, stem it and finally tokenize it into list of words. After that a custom filter is used to filter out words that are shorter than three characters, except a few that are sensitive for correct keyword detection – "c", "c#", "r", "3d", "2d", "1d", "7z", "qt".

5. Conclusion

This template can be found on the conference website <http://www.odyssey2018.org>.

6. References

[1] Stack Overflow, "Stacksample: 10% of Stack Overflow Q&A," `https://www.kaggle.com/stackoverflow/stacksample`, 2016, [Online; accessed 04-June-2018].

[2] David M. Blei, Andrew Y. Ng, and Michael I. Jordan, "Latent dirichlet allocation," *Journal of Machine Learning Research*, vol. 3, pp. 993–1022, Jan. 2003.

[3] Megan R. Brett, "Topic modeling: A basic introduction," *Journal of Digital Humanities*, vol. 2, no. 1, 2012.

[4] Anand Rajaraman and Jeffrey David Ullman, "Data mining," in *Mining of Massive Datasets*, pp. 1–17. Cambridge University Press, 2011.

[5] David M. Blei and Jon D. McAuliffe, "Supervised topic models," 2010.

[6] Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman, "Indexing by latent semantic analysis," *Journal of the American Society for Information Science*, vol. 41, no. 6, pp. 391–407, sep 1990.

[7] David M. Blei, "Probabilistic topic models," *Communications of the ACM*, vol. 55, no. 4, pp. 77, apr 2012.

[8] Chong Wang, John Paisley, and David Blei, "Online variational inference for the hierarchical dirichlet process," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, Geoffrey Gordon, David Dunson, and Miroslav Dudk, Eds., Fort Lauderdale, FL, USA, 11–13 Apr 2011, vol. 15 of *Proceedings of Machine Learning Research*, pp. 752–760, PMLR.

[9] Revi Parikh, "Garbage in, garbage out: How anomalies can wreck your data," 2014.

[10] 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*, Valletta, Malta, May 2010, pp. 45–50, ELRA, `http://is.muni.cz/publication/884893/en`.