

Report

PRODUCT DESCRIPTION GENERATOR

MEHDI BOUZIDI, ALEX ZOUTEWELLE, FURKAN UCAR

Table of contents

Problem statement.....	2
Methods	3
Data collection.....	3
Visualization	3
Pre-processing.....	8
Training.....	8
Model architecture.....	10
Results	11
Training loss.....	11
Model deployment.....	12
Tools	13
Lessons learned	14

Problem statement

Writing product descriptions can be a very time consuming activity, human written descriptions can also be unconvincing because of typos, incorrect writing style and lack of knowledge about a certain product.

This is an actual practical problem:

- An unconvincing product description can impact conversion rate, customers tend to read every letter on a product before buying it.
- Product descriptions affect SEO, the search engines rely on the use of keywords.
- Prone to human error, for example, typos or lack of knowledge about a certain product could lead to misleading the customer.
- Time consuming process usually done by content or SEO marketers.

Product Descriptions can be found on all kinds of ecommerce platforms, in many different forms. In this case we are interested in a certain range of products, along with a certain type of product descriptions. Laptops is our starting point, the data characteristics of this product category is very broad. The data usually tends to describe a certain purpose of the product. This purpose is usually linked to a certain target audience. Laptops can have a gaming purpose, meant for gamers.

Each product purpose has a different target audience, different writing style and also varies in the use of keywords. Below we compare a product descriptions with different purposes.

Product description

Multitask while you're editing a photo on the HP Pavilion Notebook 15-cw1948nd. Thanks to the 16GB of RAM, you can quickly switch between running programs. The 15.6-inch Full HD display lets you simultaneously work in 2 windows, allowing you work more efficiently in Word while browsing the internet for information. You can also bring the HP to work, as it only weighs 1.85 kilograms. The 512GB SSD lets you boot the Pavilion in 15 seconds, while also offering plenty of storage space for thousands of photos. While watching a YouTube video, you're provided with clear audio by the Bang and Olufsen speakers. On top of that, you can perfectly type in the dark thanks to the back-lit keyboard.

Editing purpose

Ready for battle and eager for a fight, the Helios 300 drops you into the game with everything you need. Only now we've armed it with NVIDIA GeForce GTX 1660 Ti graphics, the latest 9th Gen Intel Core i7 Processor and our custom-engineered 4th Gen AeroBlade 3D Fan Technology. With the 144Hz IPS panel and 3ms Overdrive response time you can say farewell to blur and hello clear, crisp, high-octane gameplay.

Gaming purpose

We see that there is a clear distinction in writing style and purpose, formal and rather informal, gaming purpose and editing purpose.

Methods

Data collection

Like any other deep learning project, this project also requires some data for the model training. So that after it gets trained it would be able to generate the descriptions of the products. We started off with the data collection of laptops. The data that we collected contains the model name, its purpose and a description that describes the purpose of the product. After that, we made a csv file of the data and then read the file using pandas data frame.

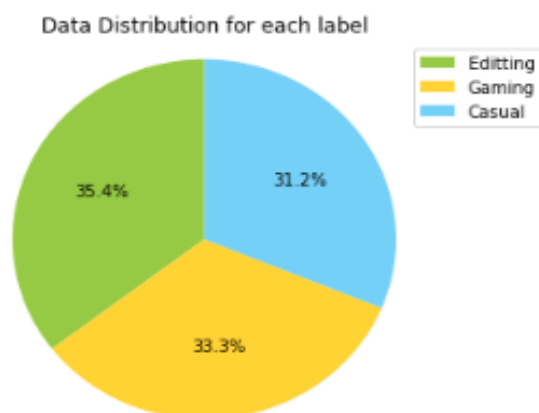
Here is the view of our sample data:

	Brand	Model	Purpose	Description
2	HP	ENVY 17-ce1906nd	Editing	On the HP ENVY 17-ce1906nd
3	HP	HP Pavilion Gaming 17-cd0917nd	Gaming	The HP Pavilion Gaming 17-cd0917nd has a 17.3-inch screen that allows you to play games such as League of Legends on a large screen on
4	Lenovo	Lenovo IdeaPad L340-17IRH 81LL003BMH	Gaming	The Lenovo IdeaPad L340-17IRH 81LL003BMH is a 17.3-inch gaming laptop with NVIDIA GeForce GTX 1650 video card. With it
5	Lenovo	Lenovo Legion Y540-17IRH 81Q4001AMH	Gaming	The Lenovo Legion Y540-17IRH 81Q4001AMH is a 17.3-inch gaming laptop that has a NVIDIA GeForce GTX 1660 Ti video card
6	HP	HP Pavilion G 17-cd0100nd	Gaming	The HP Pavilion G 17-cd0100nd is an entry-level gaming laptop for when you like to play a game of Fortnite at low settings. This is made possit
7	HP	HP 17-by2916nd	Casual	You can check your email and watch movies and series while quickly playing a game on the 17.3-inch screen of the HP 17-by2916nd laptop. Ti
8	Acer	Acer Swift 3 SF314-58-59KV	Editing	You can use the Acer Swift 3 SF314-58-59KV for your graphic studies and visual tasks. That's because this 14-inch model has an Intel Core i5
9	Acer	Acer Swift 3 SF314-57-57NU	Editing	You can use the 14-inch Acer Swift 3 SF314-57-57NU to edit photos with Photoshop while you're on the go. Thanks to the light weight of 1.19 k
10	Acer	Acer Swift 3 SF314-58G-54XQ	Editing	You can use the Acer Swift 3 SF314-58G-54XQ for your graphic tasks and visual work. The 14-inch model only weighs 1.5kg
11	Asus	Asus ZenBook UX434FLC-AI220T	Editing	Use the 14-inch Asus ZenBook UX434FLC-AI220T for all your video editing and multitasking needs. It features an Intel Core i5 processor
12	Asus	Asus ZenBook UX434FLC-AI502T	Editing	Edit photos and videos with your fingers on the touchscreen of the Asus ZenBook UX434FLC-AI502T. Thanks to the NVIDIA GeForce MX250 \

Visualization

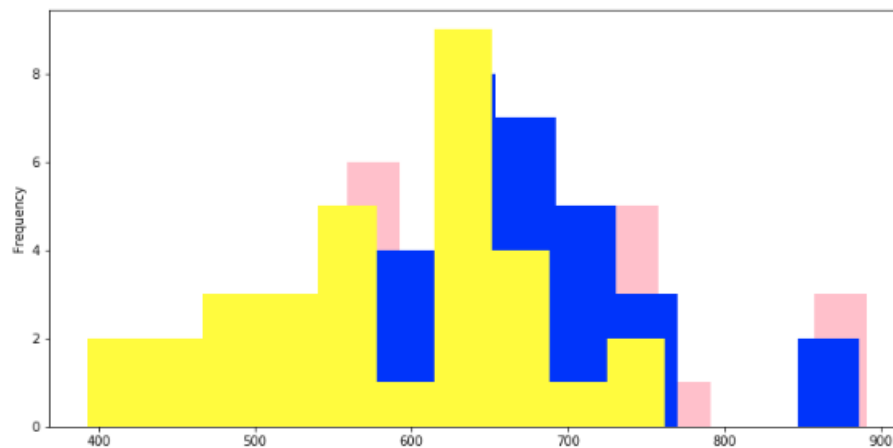
After this, we moved towards the data visualization part. Data visualization is the act of taking information (data) and placing it into a visual context, such as a map or graph. Data visualizations make big and small data easier for the human brain to understand, and visualization also makes it easier to detect patterns, trends, and outliers in groups of data. It is considered to be the vital part of any project as it shows large volumes of data in an understandable and coherent way, which in turn helps us comprehend the information and draw conclusions and insights.

For visualization we used some bar charts, histograms and word clouds to analyze the corpus that is built using our data set. We started off with checking the distribution of data for each of the product types. And for that we used a pie chart.



Then we used a histogram that shows the lengths of the description of the products with respect to frequency of the words. Histogram shows an accurate graphical representation of the distribution of data. It shows an estimate of the probability distribution of a continuous variable. Here in the figure

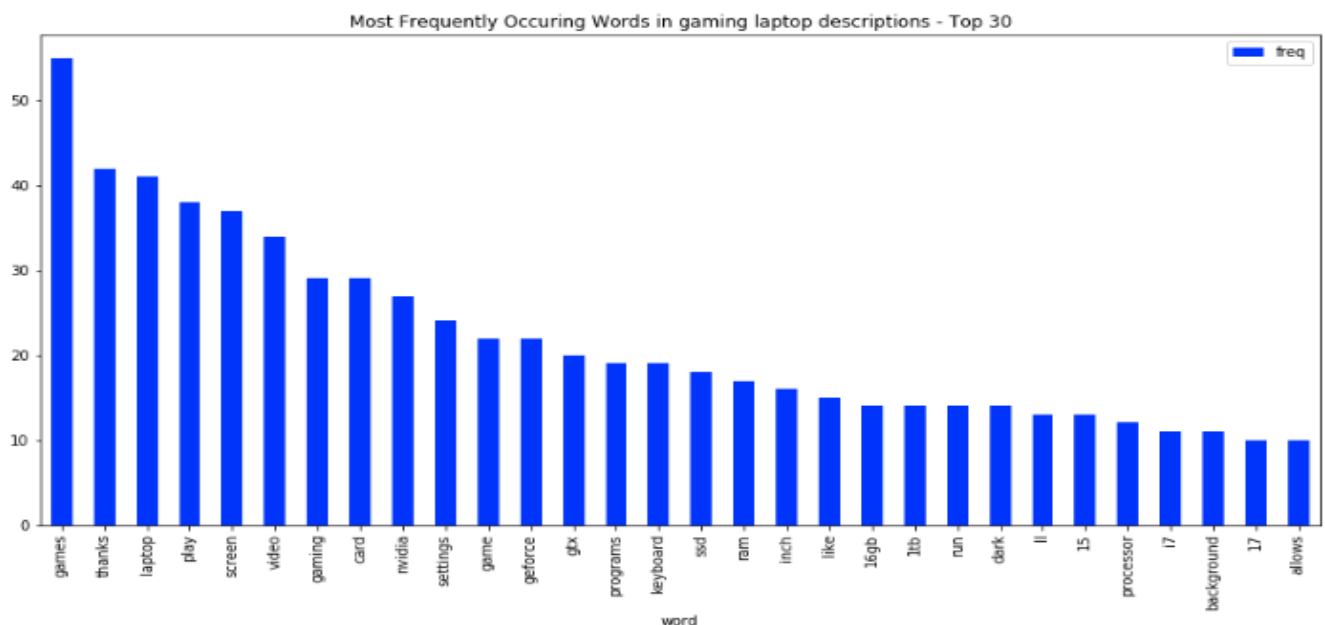
pink color is showing description of games, blue is showing the description of editing description, yellow is showing the description of casual laptops description.



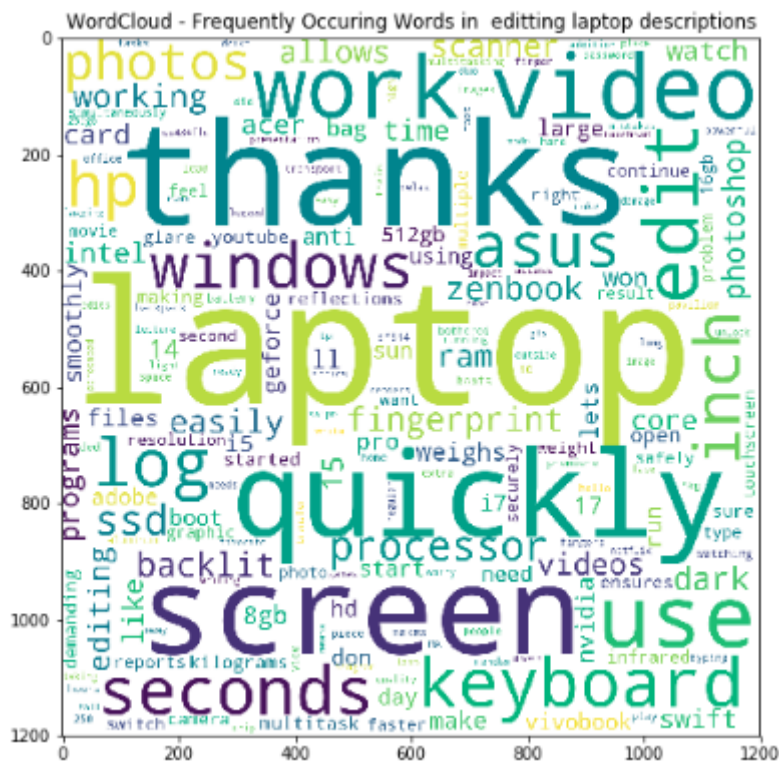
We used Word Cloud and bar chart to show the most occurring words for each product category. Word Cloud is basically a novelty visual representation of text data, typically used to depict keyword metadata (tags) on websites, or to visualize free form text. Here are word clouds and bar chart for frequently occurring words for each category.

Gaming laptop Descriptions:

Bar-chart:

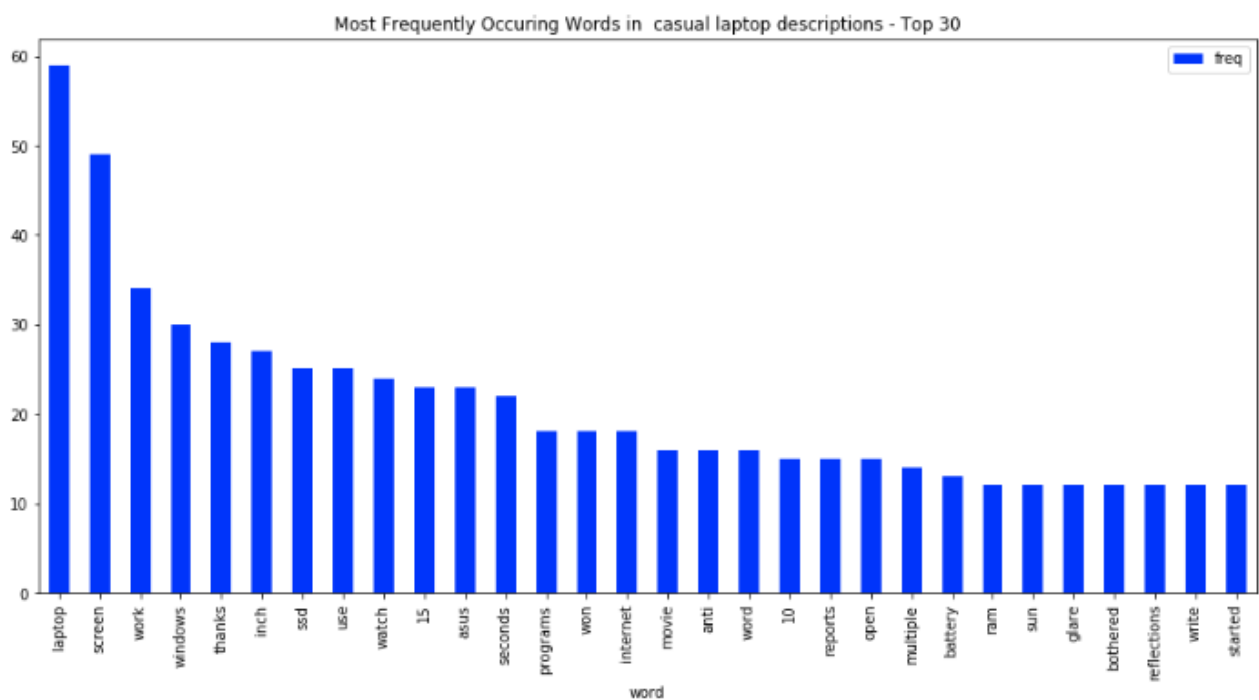


Word cloud:

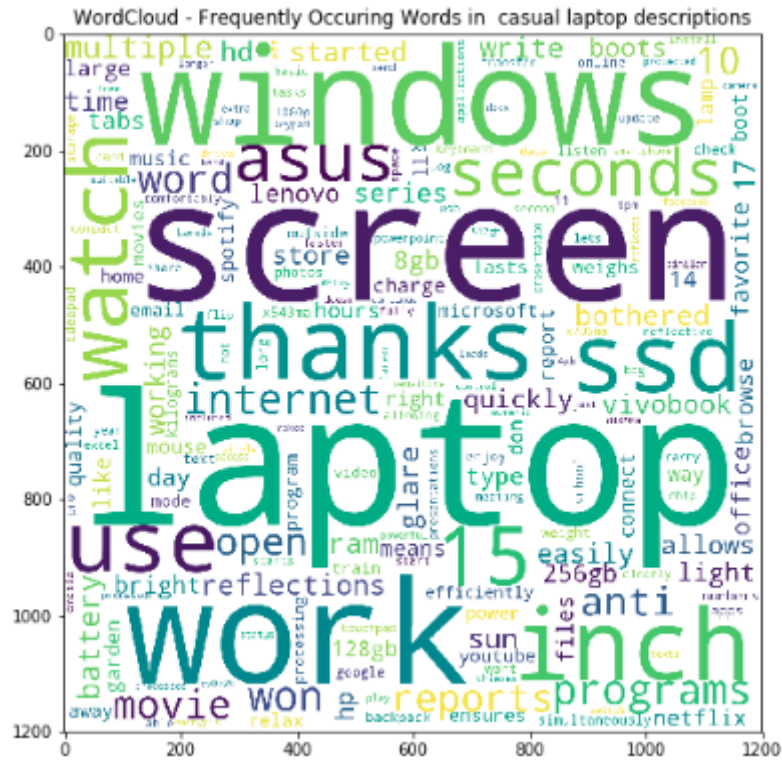


Casual laptop Descriptions:

Bar-chart:



Word cloud:



Pre-processing

Now that we have visualized our data , we will proceed towards the pre-processing of our data. Data preprocessing is a very important step for dealing with data sets. This is because it leads to better data sets, that are cleaner and are more manageable. The scraped descriptions had some flaws to them. Many of these descriptions had the brand and model name of the product in the description. In order to generate correct descriptions these model and brand names had to be filtered out. We wrote a script that filters out these brand and model names and leaves us with a clean and multipurpose description.

Training

After that we moved to the model training section. Now that we know that a model cannot work on the textual data, so our first task is to generate numeric numbers for each of the text written in the description column. So we will first store the all the sentences in the description column in an array called corpus. So for that we will first tokenize the text. Tokenization is the act of breaking up a sequence of strings into pieces such as words, keywords, phrases, symbols and other elements called tokens. Tokens can be individual words, phrases or even whole sentences. In the process of tokenization, we excluded the characters like punctuation marks “!\"#\$%&()*+,-./:;<=>?@[\\]^_`{|}~\t\n” , also we converted all the text into lowercase. Once we have tokenized our characters we convert each of the token into a sequence number. And for that we used a preprocessing function in Keras tokenizer.

We know that each description has a different amount of words, so for each line the sequence length will be different. Here are the sequences for the first 5 lines:

```
[ [8, 1],  
  [8, 1, 139],  
  [8, 1, 139, 55],  
  [8, 1, 139, 55, 2],  
  [8, 1, 139, 55, 2, 4],  
  [8, 1, 139, 55, 2, 4, 76],
```

But to feed this data in the model we need to have the same shape for each of the lines as our model cannot expect to take an input that has a different shape for each line. So for that we will make use of the padding. Since every sentence in the text has not the same number of words, we can also define the maximum number of words for each sentence, if a sentence is longer then we can drop some words. We use padding=“pre” which means it will add the zeros at the start of the sequence to make the samples in the same size. We use maxlen=155: This input defines the maximum number of words in our sentences, the default maximum length of sentences is defined by the longest sentence.

When a sentence exceeds the number of maximum words, it will drop the words and by default setting, it will drop the words at the beginning of the sentence. The result of the padding sequences is pretty straight forward. You can now observe that the list of sentences that have been padded out into a matrix where each row in the matrix has an encoded sentence with the same length, this is due to the:

- Additional zeros for short sentences and
- Truncating the sentences which exceed the max number of words which is declared by maxlen.

[illegible]

Model architecture

Now that we have all the data ready to feed into our model, we will now define a model architecture.

Layer (type)	Output Shape	Param #
embedding_1 (Embedding)	(None, 154, 10)	11950
lstm_1 (LSTM)	(None, 100)	44400
dropout_1 (Dropout)	(None, 100)	0
dense_1 (Dense)	(None, 1195)	120695
Total params: 177,045		
Trainable params: 177,045		
Non-trainable params: 0		

The model is given an input of 100 character sequences and it outputs the respective probabilities with which a character can succeed the input sequence. The model consists of 3 hidden layers. The first two hidden layers consist of 256 LSTM cells, and the second layer is fully connected to the third layer. The number of neurons in the third layer is same as the number of unique characters in the training set (the vocabulary of the training set). The neurons in the third layer, use softmax activation so as to convert their outputs into respective probabilities. LSTM layer finds and expose long range dependencies in data which is imperative for sentence structures. Dropout is a simple way to prevent overfitting in Neural Networks. A dense layer represents a matrix vector multiplication. The values in the matrix are the trainable parameters which get updated during backpropagation.

Then the model was trained for 65 epochs. We used a callback : ReduceLROnPlateau. It adjusts the learning rate when a plateau in a model's performance is detected, for example, no change for a given number of training epochs. This callback is used to reduce the learning rate after the model stops improving with the hope of fine-tuning model weights. The optimizer used was Adam. Adam is an optimization algorithm for stochastic gradient descent for training deep learning models. Adam combines the best properties of the AdaGrad and RMSProp algorithms to provide an optimization algorithm that can handle sparse gradients on noisy problems.

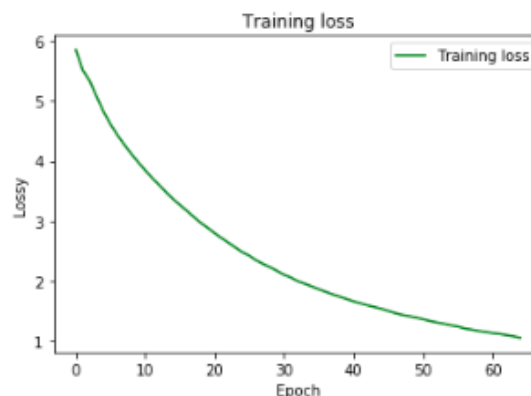
Results

We can see the descriptions as generated by the model and there are a lot of insights to conclude about the model:

- Most of the words generated by the model are proper English words, although there are exceptions at many places. This shows that the model has a good understanding of how letters are combined to form different words. Even though it is very obvious to do for a human, but for a computer model to give a reasonable performance at word formation is in itself a huge task.
- Another drawback is that the model is not able to track specific product specifications. Some data entries may consist of specific product details, for example, an HP gaming laptop, may have some characteristics which is only applicable to a MSI Gaming laptop.
- The model has some understanding of the context of the initial sentence given to it. The initial text consists of a product name along with its purpose. It then tries to make a description that is relevant to the purpose. However, sometimes in the case of a very long description it forgets that it was generating description for which purpose so it adds some content of other model.

Training loss

During training, frameworks like Keras will output the current training loss to the console. The loss is calculated as a moving average over all processed batches, meaning that in the early training stage when loss drops quickly the first batch of an epoch will have a much higher loss than the last. When the epoch is finished, the shown training loss will NOT represent the training loss at the end of the epoch but the average training loss from start to end of the epoch. So here is the progression of our training loss, we have not used a large number of epochs because that can produce a chance of overfitting the training data set.



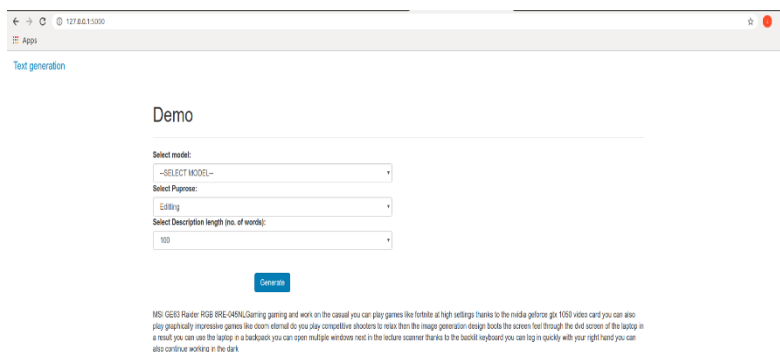
So the final loss stops at 1.05 which is considerable for the task.

Model deployment

After analyzing the results we moved towards the model deployment part. The concept of deployment refers to the application of a model for prediction using unseen identical data. Making and training the model is generally not the end of the project. Even if the purpose of the model is to increase knowledge of the data, the knowledge obtained will need to be organized and presented in a way that the customer can use it. Depending on the requirements, the deployment phase can be as simple as generating a report or as complex as implementing a repeatable data science process. In our project we keep the deployment phase as simple as possible. As a result of deployment we can generate text on the output screen.

For deploying the model we use the flask. Flask is a micro web framework which is actually written in python language. It is considered as a microframework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions.

The front end was built in HTML, CSS and Bootstrap. And for rendering the content on the web page we use simple Javascript functions. Here is the simple front end of our project. The model runs locally on the localhost 127.0.0.1. It expects the select the model name, its category, and the length of description. And as a result it produces a description of the input and shows it on the screen.



The screenshot shows a web browser window with the address bar displaying '127.0.0.1:5000'. The page title is 'Text generation'. The main content area is titled 'Demo' and contains a form with the following fields:

- Select model:** A dropdown menu with the option '-SELECT MODEL-'.
- Select Purpose:** A dropdown menu with the option 'Editing'.
- Select Description length (no. of words):** A dropdown menu with the option '100'.

Below the form is a blue button labeled 'Generate'. At the bottom of the page, there is a small block of text:

MSI GE60 Raider RGB R9E-GHML Gaming laptop and work on the casual you can play games like fortnite at high settings thanks to the midea geforce gtx 1050 video card you can also play graphically impressive games like doom eternal do you play competitive shooters to relax then the image generation design boots the screen fast through the dhd screen of the laptop in a result you can use the laptop in a backpack you can open multiple windows next in the lecture scanner thanks to the baculid keyboard you can log in quickly with your right hand you can also continue working in the dark

Tools

Since we are using Python as our main language, we chose to go along with Jupyter Notebook to implement the model. The architecture of Jupyter is language independent. The decoupling between the client and kernel makes it possible to write kernels in any language. Jupyter brings a lightweight interface for kernel languages that can be wrapped in Python. Also it allows us to create and share documents, from code to complete reports.

For the visualization of our data we used matplotlib. It is very fast and efficient. It possesses the ability to work well with many operating systems and graphic backends. It possesses high-quality graphics and plots to print and view for a range of graphs such as histograms, bar charts, pie charts, scatter plots.

With matplotlib, Keras, Sklearn all of the features worked fine. However, at first we used seaborn for the data visualization, it was visualizing data but somehow it was not easy enough to change the graph size and labels in it, so we then used matplotlib instead of it.

Lessons learned

RNN's (Recurrent Neural Networks) can also be used to generate descriptions. This means that in addition to being used for predictive models (making predictions) they can learn the sequences of a problem and then generate entirely new plausible sequences for the problem domain. So we have trained our sequence to sequence model on the corpus of laptop descriptions, which then enables our model to take a product and its category and then generate a description for it.

What surprised us was the power of LSTMs used in RNN that is more powerful in transferring relevant previous input information in the network by using a more complicated function to calculate new hidden layer neuron values. The key to their good performance is the use of cell states and different gates. So as a result, they generate quite meaningful descriptions.

We made a base model so that we can compare its performance with our RNN and it turns out that RNN was quite accurate than simple machine learning algorithms. Here are some of the results by both of the models.

Input	RNN	Baseline model (Naive Bayes)
Brand="HP " Model_name= "ENVY 17-ce1906nd " Purpose= "Editing "	HP ENVY 17-ce1906nd Editting bq731t to edit photos and watch a movie on the go of a total 10 hours with the weight ram you can easily take this laptop with your laptop or making the garden of a laptop with the anti glare screen you can store all your quickly in the garden so you won't be bothered by reflections from the sun or in a bright lamp with the anti glare screen you won't be bothered by reflections from a bright sun or lamp thanks to the backlit keyboard logging in the dark you can quickly start	HP ENVY 17-ce1906nd Editting editting editing editing editting editing editing editting editing editing editting casual casual tasks graphics 14 14 edit edit edit edit edit edit edit smooth edit edit edit edit edit smooth edit i5 vr i5 i5 i5 i5 i5 i5 6 i5 i5 i5 i5 browse internet internet inch images internet images internet internet internet images images images program images program images fluently program program only only only only fluently youtube only only youtube only only only