Monetization from a vacuum

Building a feasible online travel agency with absolutely negligible investment costs.

By Pratyush. Pradeep. Rao.

Intern at Feynn Labs,

Student of Navrachna University.

Problem Statement

There are very few people on the planet who do not want to start their own business, however, one of the barriers that stops them from actually starting one (Other than lack of motivation and other inner demons) is the investment one needs to take off, and the time and effort the business will consume to run with no guarantee of the business actually ever being successful, but what if you didn't need much initial investment and you could just start an algorithm that will run the business on cloud for you without needing much of your attention ? Something you can pursue as a side hustle without giving it much of your time, until it becomes successful and then you can start developing it ?

India is one of the biggest tourist hotspots in the world with its tourism industry worth 250 Billion USDs. With that in mind can we somehow design a travel agency that wouldn't take much of our time or effort and help small travel agencies that do not have enough money to flourish?

Business Need and Assessment

Growing up, during every vacation my family used to approach multiple travel agencies and book hotel rooms and plan trips based on their budget and the place which was most favored by the family to visit. However, the place favored by the family didn't necessarily have to be good. Suggestions provided by the travel agency were usually the popular ones, not custom ones that would cater specifically to our individual needs, ignoring the variable choices we were provided by different travel agencies.

I'm sure my family wasn't the only one to struggle with this issue. Travel Agencies are highly unreliable and biased. They will only provide you with places from which they gain the most profit from. In addition, Asking multiple travel agencies and then analyzing our options and choosing the optimal one at the end of the day is to be decided by the family itself, which is a hectic task that requires a lot of asking around and at the end of the day we aren't even sure if the place we chose would really be good as it was advertised to be because our choices would only be based on personal opinions and biases.

However, Thanks to the evolution of the internet, we have websites available that provide a platform to travel agencies to present their offers, from which the users can choose whichever one is suitable for them, based on their budget and several other factors. The problem of choosing a place that satisfies our preferences still persists. How do we decide what place would be the best for us, based on an abstract idea of the place we want to visit?

Is there a way we could quantize public opinion and quality of the place we want to visit by eliminating all the economic bias that comes from real life travel agencies, so that we can find out which place is the best for us based on our needs ? And how do we monetize on providing these suggestions ?

Target Specifications

The pipeline of the algorithm will receive three inputs from the user, the description of what kind of place the user wants to visit, their city, latitude and longitude, which in turn will yield the output as a list of optimal locations for the user, based on reach, description and public opinion.

This type of recommender system is known as Content based recommender system. We are letting the content of the place match the preference of our user.

However to provide our visitors with a content based recommendation they should have a rough idea of what they want their trip to consist of and the vicinity of the location they want to visit. In addition they should also decide how far are they willing to go for their trip which would directly be proportional to their budget.

As a compendium, this website would be targeting any user with a rough idea of what they want.

External Research

Types of recommender systems and comparing the utilization of each type:

https://www.ijert.org/research/tourism-recommendation-system-a-systematic-review-IJERTV10IS090100.pdf

Example of NLP based recommendation system for movies:

https://www.youtube.com/watch?v=1xtrIEwY_zY&list=PLKnIA16_RmvY5eP91BGPa0vXUYmIdtfPQ

Using googles Word2Vec algorithm:

https://towardsdatascience.com/a-word2vec-implementation-using-numpy-and-python-d256cf0e5f28

Hugging face hub:

https://huggingface.co/course/chapter2/1?fw=pt

Cosine Similarity:

https://www.machinelearningplus.com/nlp/cosine-similarity/

Linear Kernel:

https://towardsdatascience.com/an-intro-to-kernels-9ff6c6a6a8dc#:~:text=The%20linear%20kernel%20is%20typically,this%20kind%20of%20data%20set

Bench Marking

Tripadvisor

The original site that I took the inspiration from was tripadvisor.com the link for which you can find here.

The website is brilliant and not just for the beautiful UI, all you need to do is click on the hyperlink which will take you to the website, and from there even a toddler can figure out what to do. Search for the location or the city you want to visit and it will provide you with descriptions reviews and list of all available places that they have the data for.

There are a few problems however all linked to the fact that the website doesn't implement any AI algorithms for user convenience.

1) If the user does not have any idea about the place they want to visit, one will have to browse through multiple places and make a decision on their own which could or could not be biased to their own personal prejudices and preferences.
2) After browsing through several different places and deciding upon one, there is no guarantee if the place will actually be good, to reassure they would have to read through the reviews one by one and finalize if the place is indeed actually good.
3) Usually users won't read through all the reviews, they will read the few top ones and derive conclusions based on them alone. Knowing this websites like this keep the most positive and highest rated reviews on top so people adapt to their suggestions more often than not.

Oh if only there was some way to quantize user preferences and automate the entire process of searching and analyzing and providing recommendations that match user preferences.

Now thanks to Artificial Intelligence, there is.

Implementation of a similar Algorithm on DeepNote

While it is not exactly a web application, I came across a user that had written an end to end implementation of a content based tourist place recommendation system on deepnote. The implementation is beautiful and it works flawlessly, you can view the code here.

However a few problems with the code are as follows:

1) The code uses TfIdfVectorizer, which although isn't the worst choice a user can make for developing NLP models but it doesn't truly capture the semantic meaning of words as well as Hugging face transformers or Word2Vec algorithm developed by google, which is what I will be using in my algorithm.

2) The algorithm developed by the deepnote user does provide with excellent proficiency, a proper match for the given user input but provides no assurance of whether the place will be good or not. The sentiment of the reviews matter as well. If the place the user is looking for matches his description but isn't great then the recommendations will more likely not be accepted.

Patents and Licenses

Word2Vec: Owned by Google, is transformer trained on CBOW neural networks which used to convert words and sentences to capture their semantic meaning by storing them as vectors.

Hugging face Transformers: Owned by Hugging face, a company that provides tools frameworks and API's to trained models that can be all used in training algorithms and developing NLP models.

Applicable Regulations

Since the business idea is based on a simple algorithm and entirely online there aren't many imposed regulations from the government.

However the dataset that I've collected uses tripadvisor.com as the source collecting information about 1340 tourist places in India and it's reviews and descriptions. Since the foundation is based on tripadvisor, it might be crucial that we provide them with some credit if we decide upon monetizing this idea.

Applicable constraints

A) Abundantly available data

Unlike usual business problems availability of data isn't necessarily a problem here. There are several websites that provide with reviews and descriptions on tourist places in India. The problem here is actually the opposite. India's diversity provides with hundreds of thousands of potential tourist places and each with 20-30 reviews on an average.

To train models on such a heavy dataset and provide suitable recommendations is an industrious task and to provide suggestible recommendations it is essential to study all the possible available places and information on their vicinity. It is very likely that if this project is to be worked upon on a large scale, cloud computing services will be a must which can be a serious investment for a start up that hasn't flourished yet.

One can always start small however, for example the dataset I generated by scraping the trip advisor website consists of recommendations on about 1300 places in India which is more than sufficient and the algorithm doesn't lag on my normal PC without any internal GPU's.

B) **Lack of a reliable metric for testing**

Even if we do manage to develop a model that will provide proper recommendations there is no proper method for testing if your recommendations are accurate and effective without actually deploying the model and monitoring the changes in traffic.

There are ways to test a recommender systems validity for example Mean Average Precision metric or Collaborative filter methods but they are not as simple as the score metrics of Linear Models that we are so comfortable with so far.

Business Model

The primary question and the one you must be eager to know the answer for is how do we monetize a recommendation system ? We cannot make a subscription model because it is highly unlikely that a user would pay for simply being suggested to go to a certain place on their vacation from an AI. Vacations are not frequent and no one would pay for a one time suggestion, most people would rather just find the place themselves with a bit of a research of their own and reach to optimal conclusions. It would indeed be a very bad business model that would be doomed to fail.

However, fortunately for us that is not the idea that I came up with.

The monetary model is simple:

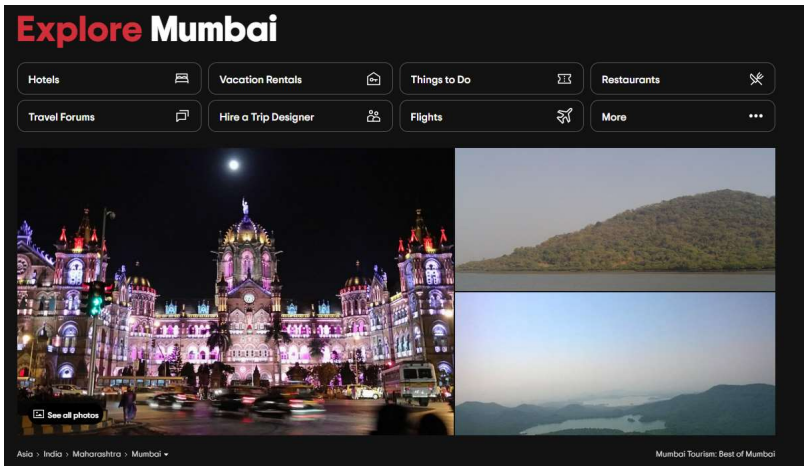
1) The website will obviously be free of cost. Users can use the website whenever convenient, as much as they would like to. Using the algorithm itself for monetary gain would be unwise and ruthless.
2) The money that we will be gaining however is by traffic, advertisements and sponsorship from Hotels. The more traffic the websites will get the more we get paid for advertising different types of brands.
3) The goal of the website is to provide the user with the optimal location to visit that matches their description of the place along with positivity and convenience of the user, it takes no responsibility in providing them with the most optimum accommodations.
4) The accommodations that we will advertise will be sponsored. As soon as the user clicks on one of our suggestions, along with the descriptions and reviews of the place we will provide them with sponsored accommodations, that they will buy and provide us with a part of the profit as commission.

The idea is simple and elegant and requires next to 0 investment cost. Except for the cloud services that might be required for the algorithm to automate and run by itself.

Concept Generation

To understand the process behind coming up with the idea we should ourselves tour the tripadvisor.com website. I as a user was browsing through it myself when I wanted to plan an urgent trip to Mumbai where my cousins live and was curious about the places one can visit in the city of sin.

When searched for Mumbai it recommends us the following page:
https://www.tripadvisor.com/Tourism-g304554-Mumbai_Maharashtra-Vacations.html



As you can see there are several options for us to choose from, there are hotels, things to do, restaurants, travel forums etc. The amount of information available on this forum is

remarkable. We have all we would need to know to correctly and thoroughly understand Mumbai, a click away, but if you view the total possible options you have.

The following link contains a list of top 30 things you can do in Mumbai:

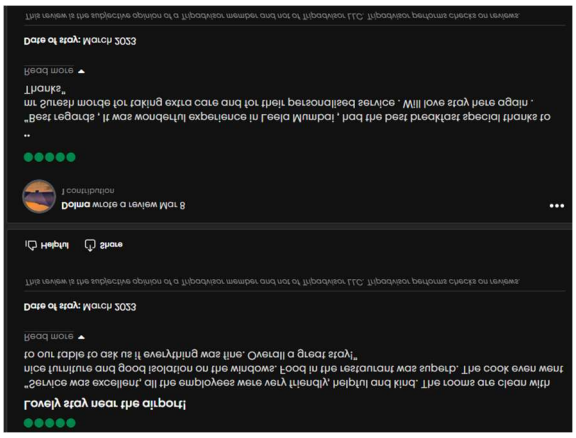
https://www.tripadvisor.com/Attractions-g304554-Activities-oa0-Mumbai_Maharashtra.html

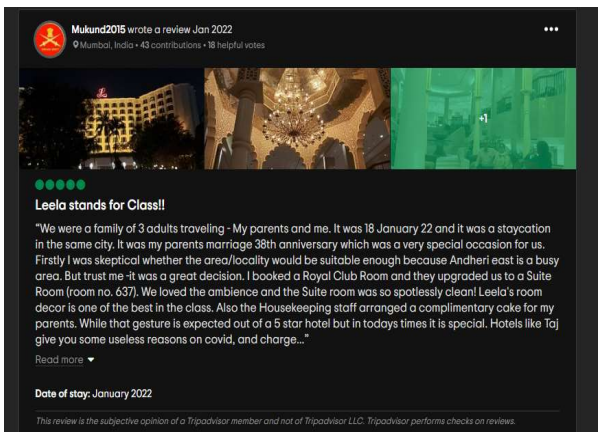
In addition this page, containing the total possible data that they have on Mumbai to do complete exploration and identify hotspots where I would have fun.

https://www.tripadvisor.com/Attractions-g304554-Activities-Mumbai_Maharashtra.html

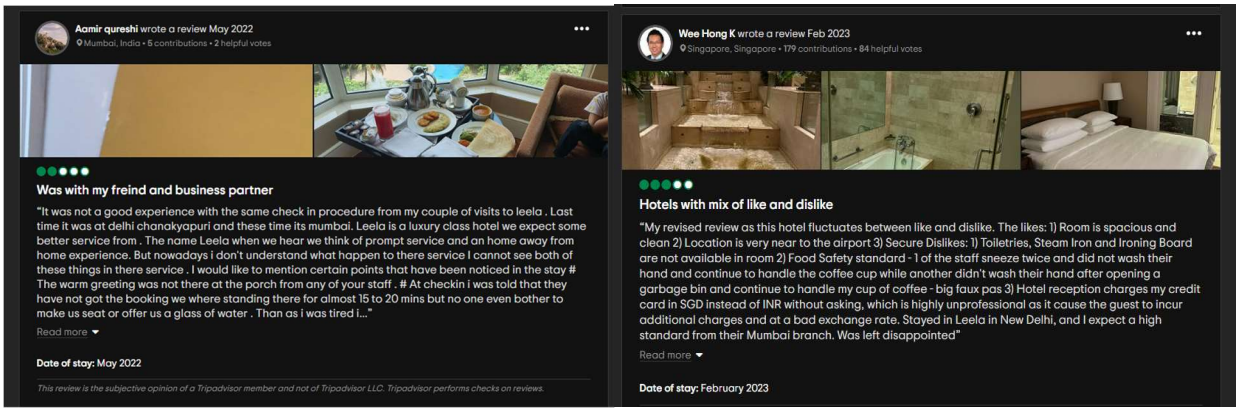
All of these places, each component with an average of 1000 reviews, and agreed Mumbai is one of the popular places in India and not every place will have this much popularity but after coming home from a tiring day, I did not have the energy to explore what Mumbai truly was. It felt like a chore that I wasn't really willing to do, I would rather put all this energy into something that I actually care about, which is when I decided that there is one way we could scrape all the information possible from this page and process it using an algorithm to pick the places for me, and even if I would be willing to do the chores myself, how much could I trust myself after an exhaustive day to make the right decision ? Even if I could manage to responsibly read through all the pages there won't be any metric that I will be comparing the places with each other, my brain can only understand if a review is positive or negative and keep count, it's hard to assign a weightage to how positive or negative a comment is, to compare the places based on actual metric without any mathematical calculations.

Moreover if you read through the reviews that you see on the pages you will see only positive reviews all over the page.





It is when you scroll through or change the sorting from most recent to most detailed you find that the real reviews which were nowhere to be seen are now right in front of you.



A machine however will have no problem reading through all of it. The good, the bad and the evil.

We are bound by our brain and body's biological limitations but computers aren't. We might not be able to do all these things but computers don't have a problem in identifying the provided metrics and reading through all possible reviews and descriptions and computing them for each place one by one using techniques in machine learning and Artificial Intelligence. This is where the idea incepted.

Concept Development with an abstract Prototype

The product of the recommendation system would be based on a Web UI and a pickled algorithm running in the background using the pre-processed data resources as it is ostensible in the provided schematic diagram.

The recommender system implemented for this product is a **Utility based recommender system** i.e. it uses user's provided input and matches it with the components of the existing dataset to provide you with the perfect utility that your desire, instead of providing recommendations based on your past visits or what type of places other customers who belong in the same cluster as you chose after visiting the previous place.

The latter would require a customer database to perform customer segmentation and implement pattern recognition algorithms to provide recommendations. This is known as **Content based recommender system.**

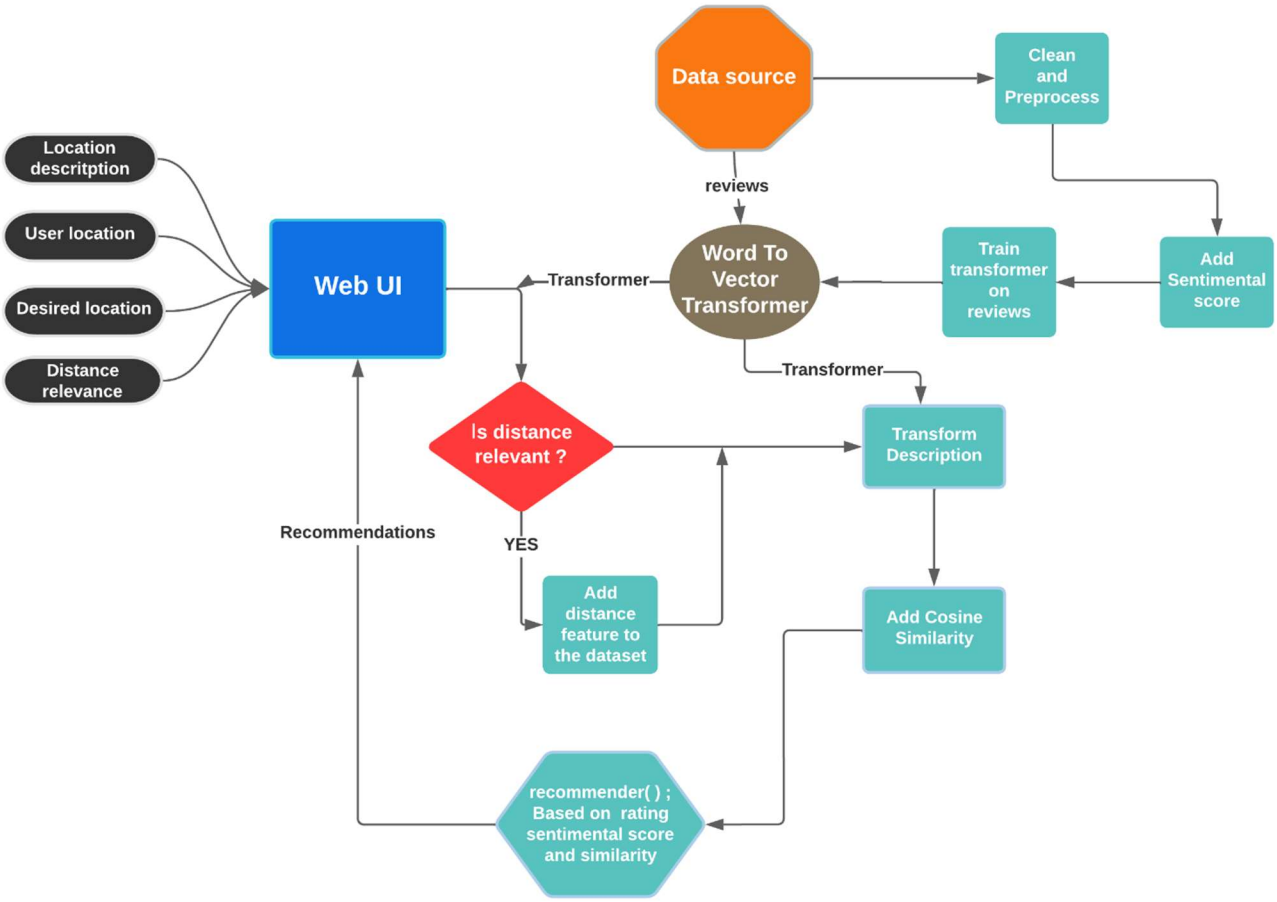
Sometimes you might want to integrate the two types to produce a recommender system that considers both. Past choices, Friends' choices and utility based. This is known as **Hybrid recommender system.**

We can either use the Word2Vec transformer trained by Google or we can use Hugging Face framework to use their provided NLP API's. Of course if we decide we want to train our personal model and we have enough available data to do so which is usually the case with functional websites, we can use gensim module in python to train our personal word to vector transformer.

The user on the client side of the server will provide with four parameters, location description, user's location, the location user desires to go at, and if distance is a relevant factor.

By matching the description with the reviews and the positive or negative sentimentality we will provide recommendations to the user based on their filtered preferences.

The recommender function at the end will provide a dataframe filtered with the most optimal recommendations along with unique id's which will then be rendered on the web UI as cards.



How does it work ?

Naturally the first step would be to train a Neural Network that will convert words to vectors in order to better encompass the semantic meaning of the individual words. But then the question arises **Why would we do that ?**

Other models like TfIdf vectorizer and Count vectorizer also are transformers that convert words into a numerical vector based on how frequently they are repeated in the sentence. So why would we use a neural network when we can avoid overcomplications ?

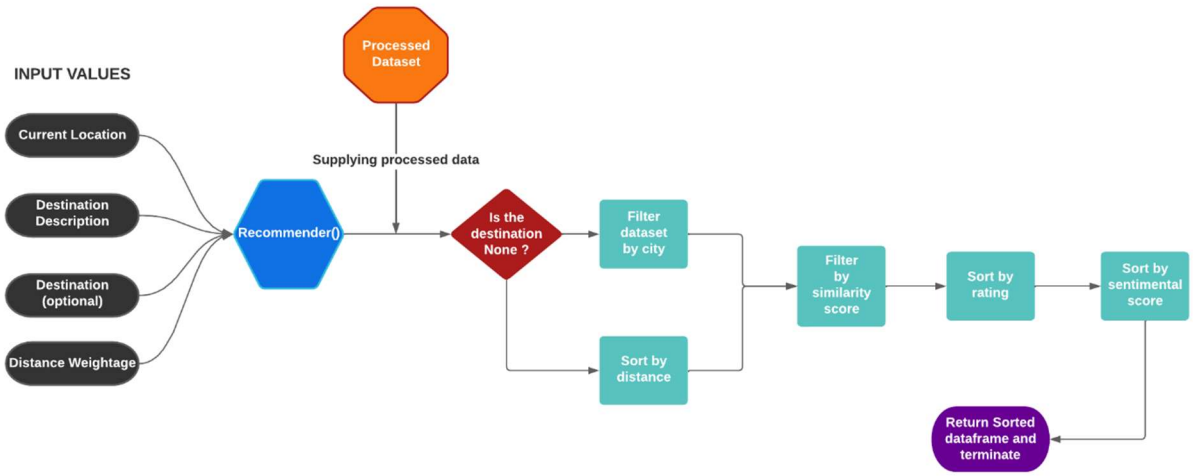
There are two problems with traditional approaches that use Bag of Words.

1) Data Sparsity: The vectors used to represent the sentences in Bag of words are sparse matrices with no limitation as to how many columns will there be.
2) Lack of Flexibility: English is a language in which one word has several adaptions. This could confuse the model while making recommendations.

The four parameters passed by the user will be the user's location, the description of the desired place and around which vicinity should the desired place be in, and whether distance is going to be in issue. The distance relevance factor will come into play when the desired city is unknown. If the city you desire is variable then we will compute distances from each city and if distance is relevant then we will consider making suggestions where the distance is a minimum. We will define a special type of city option called "None" which would allow variable optimal suggestions with minimum distance from the city you live in.

Once the inputs are passed from the client side to the server, the description will be converted into a vector which is trained on our pre-existing dataset and then we'll use utility based filtering to obtain the optimal city locations for the user.

The logic we'll follow would be this:



The recommender function would take in processed user inputs and dataframe and return a set of recommendations that will be most appropriate to the user based on their preferences.

Data sources

There is a dataset available on Kaggle that has a collection of all reviews and places of India which can be utilized for this purpose. It is a repository of around 50,000 tourist places in India with hundreds of reviews for each one of them and we can train our own personal word to vector transformer using gensim tools or Hugging face framework to feature Engineer our reviews into vectors as sentences. The dataset on Kaggle can be found here.

However my personal machine couldn't handle processing a dataset of that magnitude so instead I scraped my own using scrapy framework from tripadvisor.com to produce a dataset with 37k rows with 1300 tourist places that can be recommended.

The code for the scraping has been uploaded on my github which you can view here.

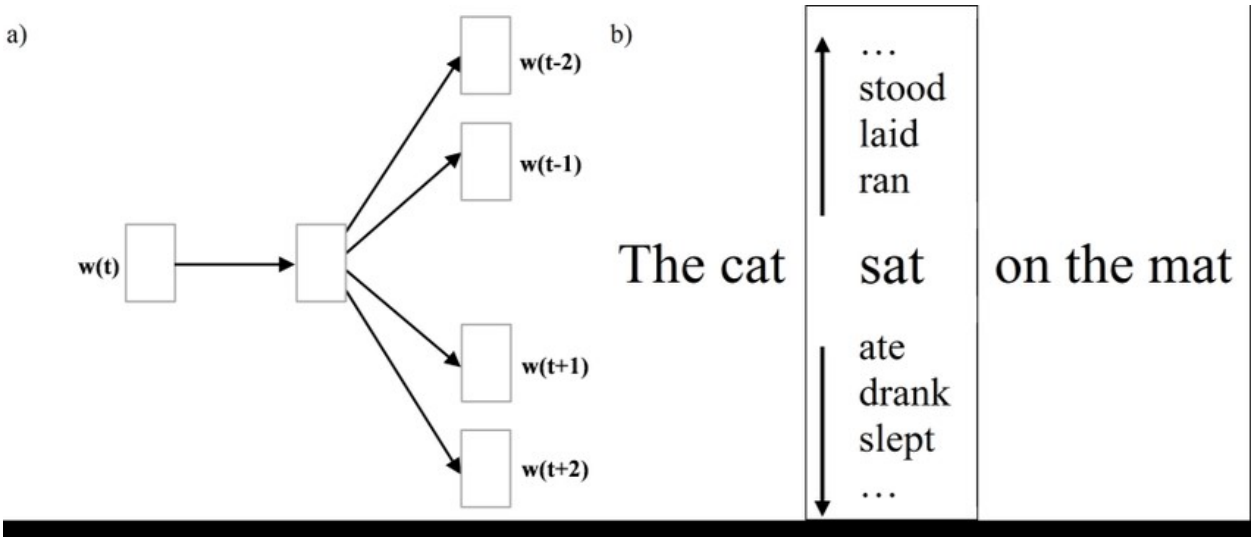
Algorithms

Word2Vec

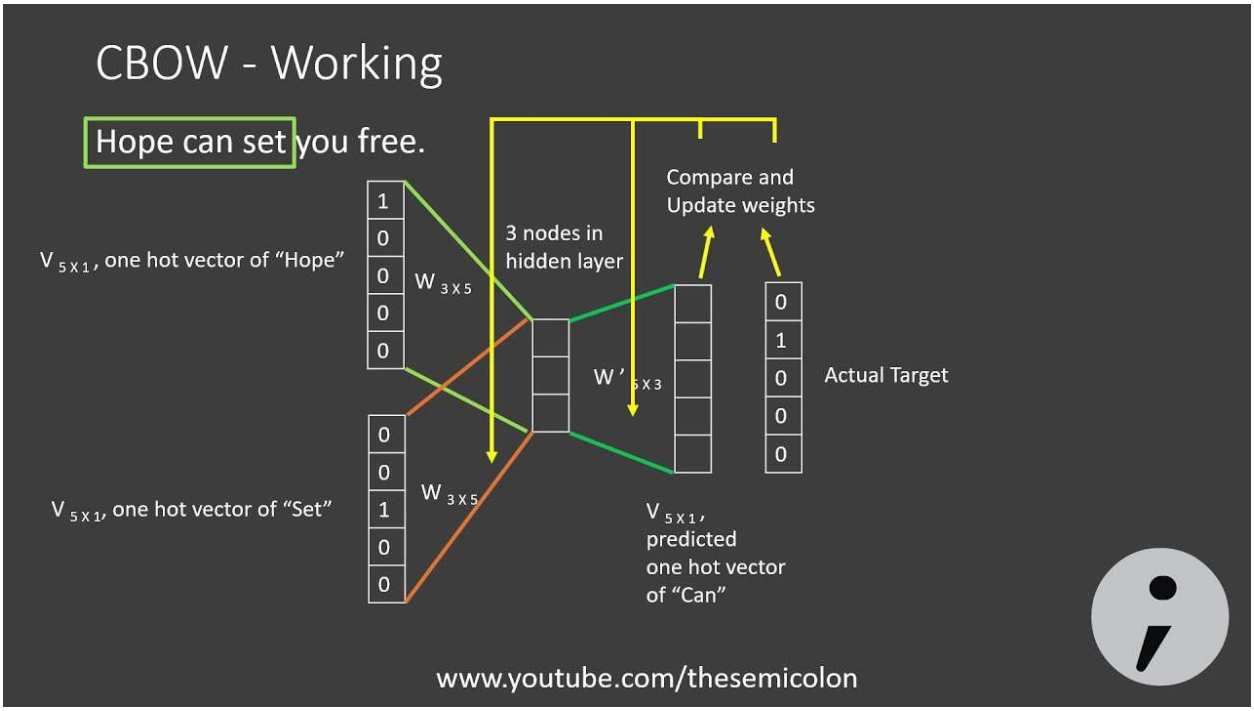
Word2vec is a combination of models used to represent distributed representations of words in a corpus C. Word2Vec (W2V) is an algorithm that accepts text corpus as an input and outputs a vector representation for each word. Here corpus is defined as the collection of all the sentences that are included in the vocabulary of the dataset.

There are two versions of this model used to produce neural word embeddings (i.e. the vectorized format of the provided word) based on the different types of neural architectures and how they learn the vectors and capture the semantic meanings.

a) Skip-Gram transformer: The skip gram neural network architecture loops over the corpus of the dataset and uses individual words as an input to predict the provided context. Everytime, the model can't predict the context words from a given input vector w it adjusts the weights of itself and tries again until it gets it right. This helps in transforming every single word in the corpus by conserving the semantic meaning of the word.



b) CBOW (continuous bag of words) : CBOW is a technique which could be thought of as an inverted version of skip-gram. This network uses the provided context of the word to predict the word in question. The architecture of this one is as follows.



If however we don't have enough data to predict the vectors we can also use Google's pre-trained Word2Vec model or BERT which another popular model used for NLP models.

However it is to be noted that skip-gram works better on large datasets.

Cosine Similarity : After converting the words to vectors we will require to compute the similarity between the description vector provided by the user and the best description available in the corpus of places. We can use cosine similarity to find out the most similar descriptions from the collections of reviews so we can find out the document that will best match our user's request.
We've all studied fundamental trigonometry to know that the angle between two vectors can be calculated using the following formula.

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}}$$

And the smaller the angle between two vectors the more parallel they are to each other. In terms of word vectors, the smaller the angle is between two vectors the more similar the words are to each other.
The cosine similarity values lie between -1 and 1 exactly like the cosine function and they tell us how similar the two words are to each other, where -1 would mean the words are complete opposites of each other and 1 would mean they have exactly the same meaning.
We'll use this property of neural word embeddings to calculate the similarity of two sentences and provide with the most similar optimal choices.

Frameworks and Team Requirements

For dealing with large datasets in python we could use pandas but it can be slow so I'd recommend using **polars** which is a modified version of pandas with higher efficiency and better analysis tools integrated along with it.

To vectorize or words we can either use google's word2vec API or the API's provided by hugging face framework that provide us with the entire set of tools we could possibly require while developing a NLP software.
However if we decide we want to train our own model because we have large enough data and we want them used in a tourism context, then we can use the

gensim package that provides us with tools we can use to train our own personal model.
We can deploy a toy website using Flask or Django, however to make a fully functional website with modern day capabilities we might need a full stack web developer who can design and add functionality to our website using Node.js, React.js, Html and CSS. We would also need sponsors and advertisers to initiate so we can provide with recommendations of their hotels along with our place recommendations.

What would it cost ?
If we were to build a basic website using flask and integrate our model to it, it would cost next to nothing but it also won't be likely to withstand the competition of the market that provide with full functionality and decadent design.
It is of course recommended that we use quality tools and design our website to withstand the test of time.
For that we would require our very own user database, in addition to a storage machine that can store and process the large magnitude of data we use to convert into vectors and compute similarity, we obviously will need to adopt cloud frameworks, luckily for us popular cloud adoption frameworks provide us with Pay As You Go service model, i.e. we only pay for what we use, if we train our model only once on the cloud and simply use the rest of it to manage user traffic and maintain a translational user database then we can pay monthly cost of a few 20-30$ a month based on the size of our traffic, since the PaaS services don't cost us for resources we don't use. It could even reach 1000's of dollars if the infrastructure we used are heavy and the traffic is high.
The website design again is a one-time investment. Once it is ready and fully functional we can integrate our model with it and provide with recommendations. With modern day utilities like Wix we can pay for maintaining a database without any management on our part and deploy our model integrated with the website. This however is not suggestible if you are aiming on establishing a full time business out of it, as it isn't very secure and there is a lot of lack of flexibility on your own data traffic.

CODE

The EDA of the provided dataset can be found on my github:
https://github.com/pickleprat/Place-Recommendation-data

Conclusions

Despite the title of the paper the idea obviously will have a few costs that have to be undertaken to establish a decently functioning business. The point I was trying to make using the title is that with only minimal costs we can produce an business which would be comparatively more effective than say opening a physical travel agency and providing suggestions simply based on your contacts and biases which could potentially lead the user to have a bad experience.

However the two worlds don't always need to collide. The algorithm is not necessarily here to take away jobs of already existing physical travel agencies.

The best way of implementing this algorithm would be by collaboration with agencies using an organizational cloud.

We can use the money of several small agencies to invest a tiny sum of money and adopt a cloud computing service to store user data in different chunks and collaborate and produce one unified effective website that will use the contacts of travel agencies with arranging a full packaged trip to a particular place and then integrate it with the website, hopefully even automating the entire procedure someday if the website becomes large enough.

_____x_____x_____