e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
Volume:03/Issue:06/June-2021 Impact Factor- 5.354 www.irjmets.com

SEMANTIC SEARCH ENGINE USING UNIVERSAL SENTENCE ENCODER

Thejeswee. N*1, Dr. V. Arun*2

*1UG Student, Department Of Computer Science And Engineering, Madanapalle Institute Of Technology & Science, Angallu (V), Madanapalle-517325, Chittoor District, Andhra Pradesh, India.

*2Assistant Professor, Department Of Computer Science And Engineering, Madanapalle Institute Of Technology & Science, Angallu (V), Madanapalle-517325, Chittoor District, Andhra Pradesh, India.

ABSTRACT

The main aim of this research is to design an efficient and low time consumption by utilizing a deep learning, advanced NLP techniques to reduce the complexity of searching. It also uses Similarity metrics like cosine similarity, set difference similarity that improves the searching process. Our main aim is to design this search engine that provides significant links with less time complexity. Users don't have to search multiple times with different words in order to get the actual results they want. This proposed search engine provides results based on the semantic meaning. Since it uses a Universal Sentence Encoder that provides semantic search [1,2], retrieval, and text clustering and it provides a balance of accuracy and inference speed [3]. We can use this work in any type of marketing and business to perform text classification, smart-reply [4] or paraphrase detection.

Keywords: Deep Learning, NLP, Universal Sentence Encoder, Vectors, Embeddings, Semantic Search, Count-Vectorizer, TF-IDF, Cosine Similarity.

I. INTRODUCTION

The Search engine using "Universal Sentence Encoder" is being developed to assist Search Engine in everyday queries in terms of reduced time spent while searching. A semantic Search engine understand the meanings hidden in retrieved user's queries. In simple terms, it is a text search where the user can have a more than English type search. Many Search engines have more latency which leads to more time in retrieving the data. Usually, Universal Sentence Encoder uses BERT model [5] which has more space & time complexity. The proposed model attempts to design an efficient and low time consumption by utilizing a deep learning, advanced NLP techniques to reduce the complexity of searching. It also uses Similarity metrics like cosine similarity that improves the searching process by: (I)A Deeper understanding of user intent. (II)A more Natural language search. (III) Understanding all the data and their context maximizes the possibility of users getting the best search experience possible. The Embeddings designed can be used to solve multiple tasks and based on the mistakes, it can be updated. As, we use the embeddings on multiple generic tasks, it discards noise and captures important features. It consists of components like Tokenization, Encoder, Multi-task Learning, Inference. Since it uses a Universal Sentence Encoder that provides semantic search, retrieval, and text clustering and it provides a balance of accuracy and inference speed. Search Engine is used to enable the user to locate & learn information on the web. This is one of the applications of deep learning. In which the users enter a query and the model does some pre-processing, cleaning, and converts into vectors, numerical vectors which is then stored as 512-dimensional vector and used to find the cosine similarity [9] between the user's searched query and the one's that is already present. Now, most similar results are displayed. The querying is done in on semantic meaning [8,9]. By doing all this we are able to maintain the application without any bugs and errors. We were able to retrieve the results in less than a second along with more accuracy in getting the similar and significant links based on semantic search of the user's query.

II. LITERATURE SURVEY

Universal Sentence Encoder [11] is a model from TensorFlow hub that encodes textual data into high dimensional vectors known as embeddings. Choosing a right Optimizer and loss function makes the model performance better in short period of time. The computational power to train the deep learning models is high and we are balancing with simple model architectures. The respective pre-trained models should be downloaded and all the packages should be installed. Once we get our required 512-dimensional vector, we use this vector to find the cosine similarity between the queries and outputs the most similar links. So, it provides the similar results in less time and more accuracy. Universal Sentence Encoder (USE) concept is used by Daniel Cer et al. [7] for similar work. Also, according to Veysel Kocaman [10], a senior data scientist has told in his article that USE model gives more accuracy when compared to other models like count-vectorizer, TF-IDF, BERT model.

III. METHODOLOGY

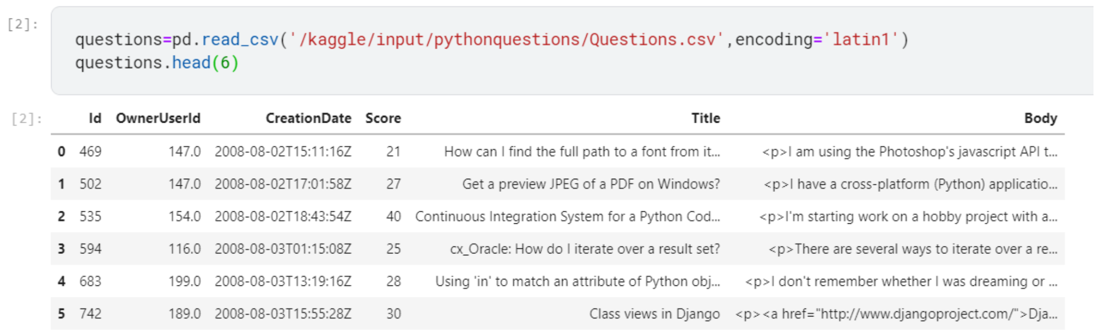
Implementation of modules

The Key functions that are to be noticed, processed and implemented are:

1. Accessing and reading "Python Questions from Stack Overflow" dataset.
2. Data loading & preprocessing.
3. Processing the Text data, cleaning.
4. Developing Universal Sentence Encoder / loading the model from TensorFlow Hub.
5. Embedding Text into Vectors (i.e., 512-dimension).
6. Calculate the cosine similarity.
7. Obtain results that are similar to search query.
8. Training the model with training dataset.
9. Evaluate the trained model with test dataset.
10. Creating a web-app with good user interface using stream lit.

In order to achieve this project, we used Universal Sentence Encoder, a model from TensorFlow hub that encodes textual data into high dimensional vectors known as embeddings. Choosing a right Optimizer and loss function makes the model performance better in short period of time. The computational power to train the deep learning models is high and we are balancing with simple model architectures. The respective pre-trained models should be downloaded and all the packages should be installed. Universal Sentence Encoder performs tokenization i.e., the sentence is converted to lowercase & are tokenized. These tokens are then passed into an Encoder that encodes sentences into 512- dimensions embedding, based on the trade-offs in accuracy vs inference this model chooses different architecture like Transformer Encoder or Deep Averaging Network (DAN) [6]. Once we get our required 512-dimensional vector, we use this vector to find the cosine similarity between the queries and outputs the most similar links. So, it provides the similar results in less time and more accuracy.

Firstly, we should input the dataset "Python Questions from stack overflow" from Kaggle, this dataset contains 3 tables – questions, answers, tags



```
[2]: questions=pd.read_csv('/kaggle/input/pythonquestions/Questions.csv',encoding='latin1')
     questions.head(6)
```

[2]:

	Id	OwnerUserId	CreationDate	Score	Title	Body
0	469	147.0	2008-08-02T15:11:16Z	21	How can I find the full path to a font from it...	<p>I am using the Photoshop's javascript API t...
1	502	147.0	2008-08-02T17:01:58Z	27	Get a preview JPEG of a PDF on Windows?	<p>I have a cross-platform (Python) applicatio...
2	535	154.0	2008-08-02T18:43:54Z	40	Continuous Integration System for a Python Cod...	<p>I'm starting work on a hobby project with a...
3	594	116.0	2008-08-03T01:15:08Z	25	cx_Oracle: How do I iterate over a result set?	<p>There are several ways to iterate over a re...
4	683	199.0	2008-08-03T13:19:16Z	28	Using 'in' to match an attribute of Python obj...	<p>I don't remember whether I was dreaming or ...
5	742	189.0	2008-08-03T15:55:28Z	30	Class views in Django	<p><a href="http://www.djangoproject.com/">Dja...

Figure 3.1: Training Dataset

After the datset is loaded and preprocessed using nlp techniques, we have all the data in required form , we can then load the model "universal Sentence Encoder" from tensor flow hub . This coverts the user input into 512-dimensional vector. This is used to find the cosine similarity between the input and existing queries. Later, this model ouputs the results with most similar sentences. All this happens in less than a second – 0.41 sec. This model can now be used to develop search engine just by adding the front-end using Stream Lit.

Psuedo code:

```
Import required packages
# load model
module_url = "https://tfhub.dev/google/universal-sentence-encoder/4"
model = hub.load(module_url)
```

print ("module %s loaded" % module_url)

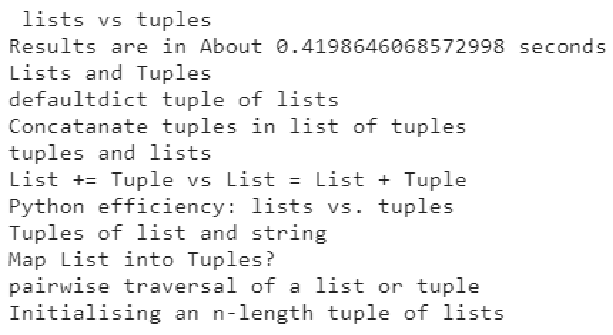
# Calculate Cosine similarity

s=np.dot(item,embed.T)

norm_a=np.linalg.norm(embed,axis=1)

norm_a=norm_a*np.linalg.norm(item)

#Print results



```
 lists vs tuples
Results are in About 0.4198646068572998 seconds
Lists and Tuples
defaultdict tuple of lists
Concatanate tuples in list of tuples
tuples and lists
List += Tuple vs List = List + Tuple
Python efficiency: lists vs. tuples
Tuples of list and string
Map List into Tuples?
pairwise traversal of a list or tuple
Initialising an n-length tuple of lists
```

Figure 3.2: Results after training the model USE

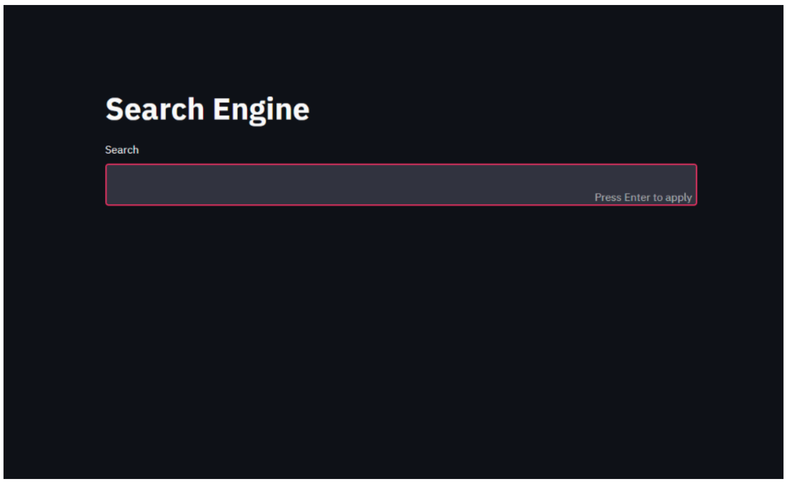
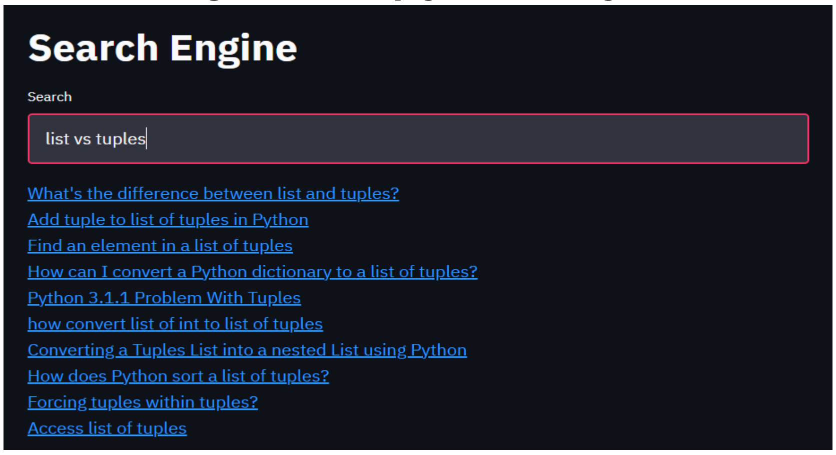


Figure 3.3: Home page for search engine



list vs tuples

[What's the difference between list and tuples?]
[Add tuple to list of tuples in Python]
[Find an element in a list of tuples]
[How can I convert a Python dictionary to a list of tuples?]
[Python 3.1.1 Problem With Tuples]
[how convert list of int to list of tuples]
[Converting a Tuples List into a nested List using Python]
[How does Python sort a list of tuples?]
[Forcing tuples within tuples?]
[Access list of tuples]

Figure 3.4: Output of search query

IV. RESULTS AND DISCUSSION

In this section, we compared the performance of existing models of search engine like count-vectorizer, TF-IDF based on the time criteria.

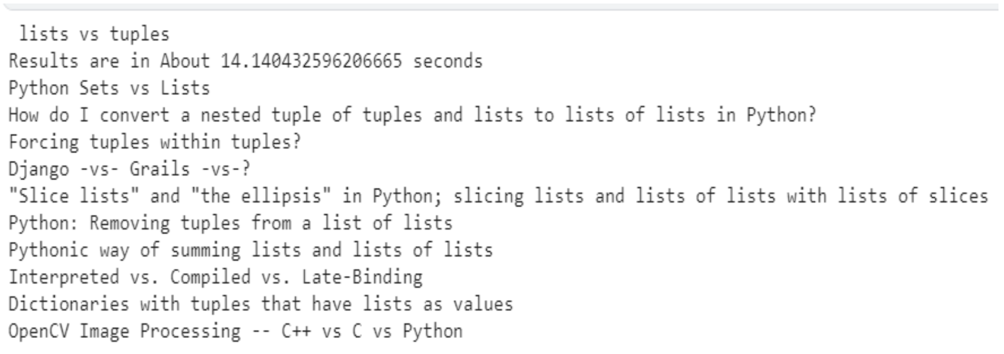
# Using NLP Model - Count vectorizer for word Representation

We import the Count Vectorizer from sklearn, this creates vector of dimensions that is very large. This model doesn't give any semantic information that leads to more trials to search for correct data/ information on web. This model takes more time to compute results also the accuracy is not up to the mark.

Pseudo code:

```
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer()
# Calculate cosine similarity
s=np.dot(item,vectors.T)
norm_a=np.linalg.norm(vectors,axis=1)
norm_a=norm_a*np.linalg.norm(item)
# Print the results
```

This model takes nearly 14.14 seconds to compute the results which is very high when compared with proposed model Universal sentence encoder.



```
 lists vs tuples
Results are in About 14.140432596206665 seconds
Python Sets vs Lists
How do I convert a nested tuple of tuples and lists to lists of lists in Python?
Forcing tuples within tuples?
Django -vs- Grails -vs-?
"Slice lists" and "the ellipsis" in Python; slicing lists and lists of lists with lists of slices
Python: Removing tuples from a list of lists
Pythonic way of summing lists and lists of lists
Interpreted vs. Compiled vs. Late-Binding
Dictionaries with tuples that have lists as values
OpenCV Image Processing -- C++ vs C vs Python
```

Figure 4.1: Results after training the model count-vectorizer

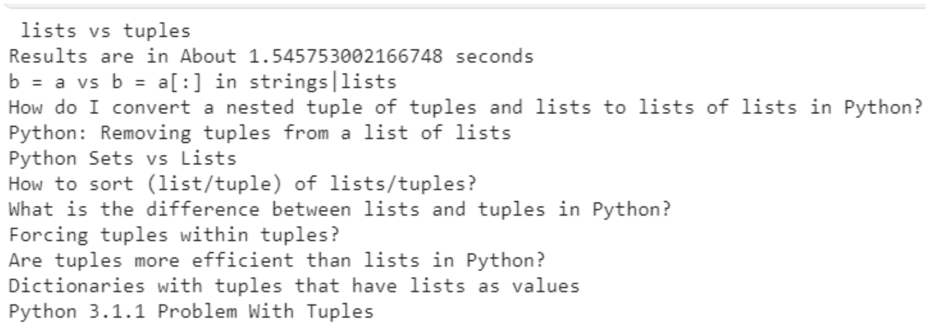
# Using TF-IDF for word representation

We import the Count Vectorizer from sklearn, this creates vector of dimensions that is very large and based on frequency of the occurrence of the word. This model gives the relevance of words in the user's input. This model takes more time to compute results when compared with USE model.

Pseudo code:

```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer()
#Calculate cosine similarity
s=np.dot(item,vectors.T)
norm_a=np.linalg.norm(vectors,axis=1)
norm_a=norm_a*np.linalg.norm(item)
#Print results
```

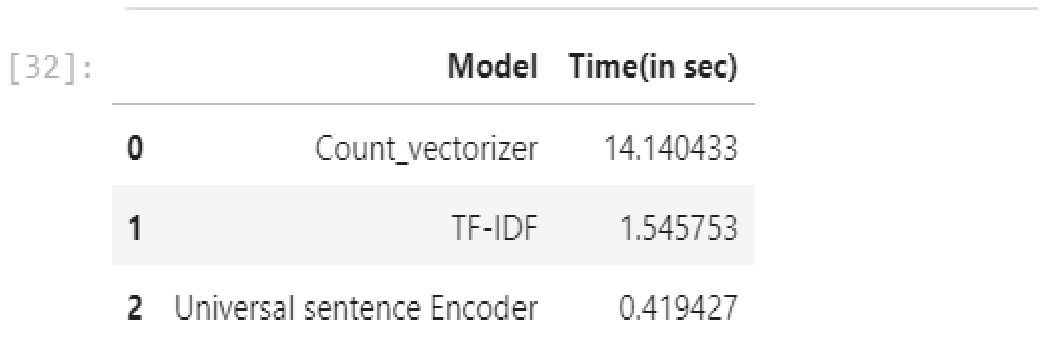
This model takes nearly 1.54 seconds to compute the results which is very high when compared with proposed model Universal sentence encoder.



```
 lists vs tuples
Results are in About 1.545753002166748 seconds
b = a vs b = a[:] in strings|lists
How do I convert a nested tuple of tuples and lists to lists of lists in Python?
Python: Removing tuples from a list of lists
Python Sets vs Lists
How to sort (list/tuple) of lists/tuples?
What is the difference between lists and tuples in Python?
Forcing tuples within tuples?
Are tuples more efficient than lists in Python?
Dictionaries with tuples that have lists as values
Python 3.1.1 Problem With Tuples
```

Figure 4.2: Results after training the model TF-ID

# Model performance comparison



[32]:

	Model	Time(in sec)
0	Count_vectorizer	14.140433
1	TF-IDF	1.545753
2	Universal sentence Encoder	0.419427

Figure 4.3: Comparison of models based on time taken to produce results

```
[33]:  sns.set(style="whitegrid")
       ax = sns.barplot(y="Model", x="Time(in sec)", data=pc)
```

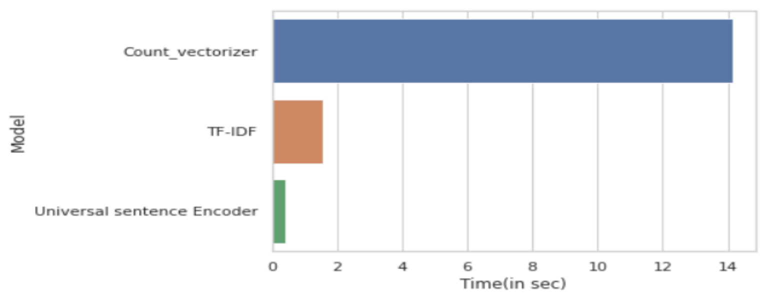


Figure 4.4: Bar plot of models based on time taken

Hence the proposed model is easy to use, with low-computational power and does not require any special training. A user can simply search the queries on the web-based application and due its semantic search it makes easier to find the required similar results in less than a second. When we compare this with other existing models it proves that universal sentence encoder provides results in less time (0.41 seconds) also, the resultant links are much similar and accurate to the search query. As a whole, the system is just simplified and scalable. Due to the use of NLP and deep learning models this is well trained model that gives accurate results in less time. While validating the model with test data the results were accurate and well-defined with semantic meaning.

V. CONCLUSION

The conclusion of this work is the comparative result of semantic search with less computational time and high accuracy. Hence from the above proposed method we have designed a search engine using Universal Sentence Encoder that is capable to produce results in less time with more accuracy. The proposed model was designed with encoder architecture. Universal Sentence Encoder is a model from TensorFlow hub that encodes textual data into high dimensional vectors known as embeddings, based on the trade-offs in accuracy vs inference this model chooses different architecture like Transformer Encoder or Deep Averaging Network (DAN). It goes through multi-task learning and adds the semantic meaning to the query. Once the model is trained then we can use for semantic searches, paraphrase detection, smart-reply, text classification. Once we get our required 512-dimensional vector, we use this vector to find the cosine similarity between the queries and outputs the most similar links. So, it provides the similar results in less time and more accuracy. The model is fully capable of being trained by using the

stochastic gradient descent that makes the training process easier. The experimental evaluations indicate that the proposed model is able to generate more accurate results automatically.

The proposed model attempts to design an efficient and low time consumption by utilizing a deep learning, advanced NLP techniques to reduce the complexity of searching. We can achieve more than 90% validation accuracy in seconds.

VI. REFERENCES

[1] Si, S., Zheng, W., Zhou, L., Zhang, M.: Sentence similarity computation in question answering robot. IOP Conf. Ser. J. Phys. Conf. Ser. **1237**, 022093 (2019)Google Scholar

[2] Jeon, J., Bruce Croft, W., Lee, J.H.: Finding semantically similar questions based on their answers (Copyright is held by the author/owner. SIGIR'05, August 15–19, 2005, Salvador, Brazil)Google Scholar

[3] Alexis Conneau et al., "Supervised Learning of Universal Sentence Representations from Natural Language Inference Data"

[4] Matthew Henderson et al., "Efficient Natural Language Response Suggestion for Smart Reply"

[5] Reimers, N., Gurevych, I.: Sentence-BERT: sentence embeddings using Siamese BERT-networks. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics (2019). https://arxiv.org/abs/1908.10084

[6] https://amitness.com/2020/06/universal-sentence-encoder/.

[7] Daniel Cer et al., "Universal Sentence Encoder"

[8] Yinfei Yang et al., "Learning Semantic Textual Similarity from Conversations"

[9] Google AI Blog, "Advances in Semantic Textual Similarity"

[10] https://towardsdatascience.com/text-classification-in-spark-nlp-with-bert-and-universal-sentence-encoders-e644d618ca32

[11] Cer, D., et al.: Universal sentence encoder. arXiv preprint arXiv:1803.11175 (2018).