Uncovering Key Drivers in Selecting Professionals for Global Software Development and the Gig Economy

Erivelton Antonio dos Santos, Dalton Garcia Borges de Souza, and Carlos Eduardo Sanches da Silva

This document contains essential **Supporting Information** concerning the **Algorithms** presents in the paper.

APPENDIX A
CCSS ALGORITHM

This Appendix A presents the criteria clusters' semantic similarity (CCSS) algorithm1 used in Python framework.

APPENDIX B
ALGORITHM FOR THE 3D BUBBLE CHART

This Appendix B presents the algorithm 2, an interactive 3D hierarchical clustering graph illustrating the hierarchical structure of CC.

Algorithm 1 CCSS between the clusters

```
# Data Structures
import numpy  as np
import pandas as pd

# Keyword extraction
from sklearn.feature_extraction.text import CountVectorizer
from sentence_transformers import SentenceTransformer
from sklearn.metrics.pairwise import cosine_similarity

# Loading the data
df = pd.read_excel('Clusters definitions_vf01.xlsx')
df.head()

# Importing the clustering algorithms - initialize our model and tokenizer
model_name = 'sentence-transformers/all-mpnet-base-v2'
from transformers import AutoTokenizer, AutoModel
import torch
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModel.from_pretrained(model_name)

# Declaring the variables - tokenize the sentences
sentences = df['Cluster definition']
sentences
sentences_list = list(df['Cluster definition'])
sentences_list
tokens = {'input_ids': [], 'attention_mask': []}
for sentence in sentences_list:
    new_tokens = tokenizer.encode_plus(sentence, max_length=384,
                                       truncation=True, padding='max_length',return_tensors='pt',
                                       return_attention_mask=True)

    tokens['input_ids'].append(new_tokens['input_ids'][0])
    tokens['attention_mask'].append(new_tokens['attention_mask'][0])

tokens['input_ids']

# reformat list of tensors into single tensor
tokens['input_ids'] = torch.stack(tokens['input_ids'])
tokens['attention_mask'] = torch.stack(tokens['attention_mask'])

# Checking the variables
tokens['input_ids']
type(tokens['input_ids'])
tokens['input_ids'].shape

# Making the operations - Processing these tokens through our model
outputs = model(**tokens)
outputs.keys()

# The dense vector declarations of our text are contained within the outputs 'last_hidden_state' tensor
embeddings = outputs.last_hidden_state
embeddings
embeddings.shape

# Resize our attention_mask tensor
attention = tokens['attention_mask']
attention.shape
mask = attention.unsqueeze(-1).expand(embeddings.shape).float()
mask

# Multiply the two tensors to apply the attention masks
mask_embeddings = embeddings * mask
mask_embeddings
mask_embeddings.shape

# Then we sum the remained of the embeddings along axis 1
summed = torch.sum(mask_embeddings, 1)
summed.shape
summed

# Sum the number of values that must be given attention in each position of the tensor
counts = torch.clamp(mask.sum(1), min=1e-9)
counts.shape
counts

# Calculate the mean as the sum of the embedding activation's summed divided bythe number of values that should be given attention in each position counts
mean_pooled = summed / counts
mean_pooled.shape
mean_pooled

# The final operations - calculate the cosine similarity between the vectors
from sklearn.metrics.pairwise import cosine_similarity
mean_pooled = mean_pooled.detach().numpy()
data_25g = cosine_similarity(
[mean_pooled[0]],
mean_pooled[1 :]
)
data_25g # data_25g is the final similarities matrix.

#Print to spreadsheet
data_25gT = data_25g.T
data_25gT
dfdata_25gT = pd.DataFrame(data_25gT)
dfdata_25gT.to_excel("saida.xlsx", index=False)
```

Algorithm 2 Algorithm for the 3D bubble chart

```
import plotly.express as px
import numpy as np
import pandas as pd
```

```
df = pd.read_excel('file_3D.xlsx')
df.head()
```

	Cluster	Description	Ri + Di	Ri - Di	Cosine Sim	W_cluster
0	COMMUN	Communication	5.42	0.43	1.000000	1
1	INTERF	Component interface	4.69	0.16	0.333691	2
2	SCIENT	Scientific attitude	3.91	0.25	0.332624	2
3	METRIC	Quality metrics	3.69	0.04	0.213171	2
4	GREENSO	Green software development	3.69	0.01	0.212376	2

```
fig = px.scatter_3d(df, x='Ri - Di', y='Ri + Di', z='Cosine Sim', size='Ri + Di', color='W_cluster',
                hover_data=['Description'])
fig.update_layout(scene_zaxis_type="log")
fig.show()
```