



## ΤΕΛΙΚΗ ΑΝΑΦΟΡΑ ΠΡΟΟΔΟΥ ΠΡΑΚΤΙΚΗΣ ΑΣΚΗΣΗΣ

**Όνομα ασκούμενου:** Παπαγεωργίου Μάριος

**Αρ. Μητρώου:** 3190156

**Φορέας Υλοποίησης Π.Α.:** ΑΓΑΠΑΕΙ ΠΛΗΡΟΦΟΡΙΚΗ Ε.Π.Ε.

**Επιβλέπων Καθηγητής:** Δρ. Καλογεράκη Βασιλική

**Εργασιακός Επιβλέποντας:** Τσάμης Νικόλαος

**Ακαδημαϊκό Εξάμηνο:** Καλοκαιρινή Περίοδος - Ακ. Έτος 2022 - 2023

Αθήνα, 3/10/2023



Επιχειρησιακό Πρόγραμμα  
Ανάπτυξη Ανθρώπινου Δυναμικού,  
Εκπαίδευση και Διά Βίου Μάθηση

Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης



## Πίνακας περιεχομένων

<b>Περιγραφή της εταιρείας.....</b>	<b>2</b>
<b>Νέες γνώσεις και ένταξη στον επαγγελματικό τομέα.....</b>	<b>3</b>
<b>Στόχος.....</b>	<b>3</b>
<b>Ρόλοι και καθήκοντα.....</b>	<b>3</b>
<b>Ανάλυση Καθηκόντων.....</b>	<b>4</b>
Eventora Smart Invitation Productor.....	4
Εισαγωγή.....	4
OpenAI.....	4
Backend Python.....	5
Backend C#.....	6
Full Stack ASP NET 6.0 Framework.....	7
Τι πετύχαμε.....	10
Eventora Spread The Word.....	10
Εισαγωγή.....	10
Long Short Term Memory.....	10
Sentiment Analysis.....	11
Κατασκευή μοντέλου μέσω Google Colab.....	11
Χρήση του μοντέλου σε Python.....	13
Χρήση του μοντέλου σε C#.....	14
Full Stack ASP NET 6.0 Framework.....	15
Χρήση του μοντέλου για ελληνικά σχόλια.....	18
Amazon Web Services (AWS).....	20
Πρόσβαση δεδομένων μέσω AWS.....	20
Κατασκευή και χρήση μεθόδων μέσω AWS.....	21
Sagemaker.....	21
Κατασκευή μοντέλου μοντέλου Sagemaker.....	22
Τι πετύχαμε.....	25
<b>Πηγές γνώσεων.....</b>	<b>26</b>
<b>Προβλήματα.....</b>	<b>26</b>
Διακοπή ρεύματος.....	26
Σύνδεση στο διαδίκτυο.....	26
Συσκευή.....	27
<b>Επίλογος.....</b>	<b>27</b>
<b>Χρήσιμοι σύνδεσμοι.....</b>	<b>27</b>

Η συνεργασία μου με την εταιρεία **ΑΓΑΠΑΕΙ ΠΛΗΡΟΦΟΡΙΚΗ** αποδείχθηκε ιδιαίτερα ικανοποιητική, τόσο για μένα όσο και για το προσωπικό. Μάλιστα προς το τέλος της πρακτικής μου, ο επιβλέπων μου στην εταιρεία μου έκανε πρόταση να δουλέψω μαζί τους στο μέλλον, αφότου έχω ολοκληρώσει τις σπουδές μου και αναζητώ θέση εργασίας.

## **Περιγραφή της εταιρείας**

Η ΑΓΑΠΑΕΙ ΠΛΗΡΟΦΟΡΙΚΗ είναι η δημιουργός της καινοτόμας βραβευμένης πλατφόρμας **Eventora** η οποία βοηθάει εταιρείες να οργανώσουν εκδηλώσεις διά ζώσης, online, και υβριδικές όπως συνέδρια, εκθέσεις, διαγωνισμούς, ημέρες καριέρας, και επιχειρηματικές συναντήσεις.

Αναπτύχθηκε στην Ελλάδα, αλλά έχει επεκταθεί και σε παγκόσμιο επίπεδο. Έχει συνεργαστεί με πολλές μεγάλες εταιρείες, τόσο εντός της Ελλάδας όσο και στο εξωτερικό, όπως η Aegean, Coca-Cola, ΔΕΗ, Helleniq Energy, Eurobank, EY, NN, Παπαστράτος, Τράπεζα Πειραιώς, Protergia, TEDx και πολλές άλλες.

Η επίσημη ιστοσελίδα [www.eventora.com](http://www.eventora.com) παρέχει τη δυνατότητα κατασκευής διαφόρων στοιχείων εκδήλωσης όπως προσκλήσεις, μηνύματα εγγραφής και ειδοποιήσεων, εισιτήρια, ερωτηματολόγια, κληρώσεις, ψηφοφορίες, δημοσκοπήσεις και οτιδήποτε άλλο επιθυμεί κάθε διοργανωτής.



## **Νέες γνώσεις και ένταξη στον επαγγελματικό τομέα**

Η πρακτική άσκηση ήταν το κατάλληλο βήμα για να ενταχθώ στον επαγγελματικό κόσμο της πληροφορικής.

Με βοήθησε να δω πως μπορούν να χρησιμεύσουν οι γνώσεις που απέκτησα όσο καιρό βρίσκομαι στο πανεπιστήμιο και ότι μπορούν να επεκταθούν ακόμα περισσότερο.

Αξίζει να σημειωθεί πως απέκτησα ακόμη περισσότερες γνώσεις και ικανότητες που δεν είχα διδαχθεί προηγουμένως, που πλέον μπορώ και να συνδυάσω με αυτές που είχα εξαρχής.

## **Στόχος**

Κύριο θέμα που ο εργασιακός επιβλέπων μου είχε υπόψη ήταν να βρούμε ιδέες τις οποίες θα μπορούσα να υλοποιήσω ούτως ώστε αφενός να με βοηθήσουν να επεκτείνω τις ικανότητες μου στο προγραμματισμό, αφετέρου να παράγω έργο που θα επεκτείνει τις δυνατότητες της Eventora.

Οι ιδέες που εν τέλει πήραν μορφή εστιάζουν στην χρήση της Τεχνητής Νοημοσύνης και της Μηχανικής Μάθησης, δύο τομείς οι οποίοι εξελίσσονται συνεχώς και έχουν όλο και μεγαλύτερη απαίτηση.

## Ρόλοι και καθήκοντα

Βασικός μου ρόλος στην ομάδα που εντάχθηκα ήταν full-stack-developer για Web Applications. Βέβαια, ο κώδικας που έγραψα βρίσκεται σε project ξεχωριστό από την επίσημη πλατφόρμα της Eventora. Προτεραιότητα ήταν να κατασκευάσω, να ελέγξω και να παραδώσω ένα project το οποίο θα μπορούσε εύκολα να κατανοηθεί, να αναδιαμορφωθεί και να ενταχθεί στην πλατφόρμα ως επιπρόσθετη λειτουργία.

Υστερα από αρκετή έρευνα, κατέληξα σε δύο ιδέες τις οποίες παρουσίασα στον επιβλέπων μου. Τις έγκρινε με ενθουσιασμό καθώς συμφωνήσαμε πως θα ενισχύσουν την εικόνα της Eventora και θα ανοίξει νέους ορίζοντες για ακόμη περισσότερες υπηρεσίες.

Τα τελικά web applications υλοποιήθηκαν σε ένα ASP.NET Framework project με χρήση του εργαλείου Visual Studio Code Community Edition με γλώσσα C#. Οι δύο αυτές ιδέες με δική μου πρωτοβουλία ονομάστηκαν:

1. **Eventora Smart Invitation Productor (S.I.P.)**
2. **Eventora Spread The Word (S.T.W.)**

## Ανάλυση Καθηκόντων

Παρακάτω παρουσιάζονται ξεχωριστά οι αφορμές, οι στόχοι, και όλα τα βήματα που στο τέλος έδωσαν μορφή στα web applications.

### Eventora Smart Invitation Productor

#### Εισαγωγή

Η εταιρεία έχει συνεργαστεί στο παρελθόν με διοργανωτές οι οποίοι έχουν δώσει μόνο κάποια βασικά στοιχεία μιας εκδήλωσης και έχουν ζητήσει από την ίδια να συντάξει τις προσκλήσεις και τα μηνύματα εγγραφής. Αυτό είναι επιβάρυνση για την εταιρεία, επειδή για έναν πελάτη χρειάζεται επιπλέον χρόνος για σύνταξη και διόρθωση συντακτικών λαθών.

Το **Eventora S.I.P.** παρέχει λύση σε αυτό το πρόβλημα. Στόχος του είναι να δημιουργήσει με τη βοήθεια της Τεχνητής Νοημοσύνης προσκλήσεις και μηνύματα εγγραφής χρησιμοποιώντας μόνο βασικά στοιχεία μιας εκδήλωσης και να παρέχει επιπλέον δυνατότητα στους χρήστες να τις τροποποιούν και να τις αποθηκεύουν.

#### OpenAI

Το κομμάτι της Τεχνητής Νοημοσύνης βασίζεται στην πλατφόρμα OpenAI, δημιουργό της πασίγνωστης πλατφόρμας ChatGPT. Για τον κώδικα του Eventora S.I.P. αξιοποιήθηκε ένα μοντέλο GPT 3.5 turbo. Για τη χρήση του απαιτείται:

- Ένας λογαριασμός στο OpenAI (κάθε νέος λογαριασμός έχει 3 μήνες free trial με 15\$ budget) και

- Ένα API κλειδί που δημιουργούμε στην ιστοσελίδα (και πρέπει να έχουμε αποθηκευμένο σε κάποιον δευτερεύοντα χώρο στη συσκευή μας, π.χ. αρχείο .txt)

## Backend Python

Αρχικά κατασκεύασα έναν κώδικα Python που με ένα απλό μενού επιλογών και συμπλήρωση κειμένου παράγει προσκλήσεις και μηνύματα και μπορεί να τροποποιήσει με συγκεκριμένα φίλτρα.

Σε κάθε επιλογή του μενού αποστέλλουμε στο μοντέλο GPT ένα ερώτημα, prompt και για αυτό παράγεται μία απάντηση εντός ολίγων λεπτών. Όλα τα ερωτήματα και οι απαντήσεις αποθηκεύονται ως μηνύματα σε λίστα, σαν μία απλή συνομιλία μεταξύ του χρήστη και του μοντέλου.

```

188 def generate_answer(my_conversation):
189     print('Writing...') # print message to make sure the answer is generating
190     try:
191         # response generated for the AI
192         response = openai.ChatCompletion.create(model=model_id, messages=my_conversation, frequency_penalty=0.2,
193                                                 presence_penalty=0.8)
194         # add the message in chat
195         my_conversation.append(
196             {'role': response.choices[0].message.role, 'content': response.choices[0].message.content})
197
198     return my_conversation
199 except Exception as e:
200     print('Error during generating: ', e)
201
202
203 conversation = [{'role': 'system', 'content': 'How may i help you?'}]
204 conversation = generate_answer(conversation)
205 print('{0}: {1}\n'.format(*args: conversation[-1]['role'].strip(), conversation[-1]['content'].strip()))
206
207 # Logic of the chat for the event creation:
208 # 1. We select all mandatory and already existent fields
209 # 2. We give the event's name
210 # 3. We give one description and some comments in order to let the program create a more professional invitation
211 # 4. After that the rest work like ChatGPT
212 # !Note: The chat doesn't last forever! There is a specific amount of tokens
213 # When all of them are consumed the program ends
214
215 prompt = 'I want to make an invitation for a ' + filter_category + ' event named ' + filter_name + ' For more details i give'
216 conversation.append({'role': 'user', 'content': prompt})
217 conversation = generate_answer(conversation)
218 print('{0}: {1}\n'.format(*args: conversation[-1]['role'].strip(), conversation[-1]['content'].strip()))
219

```

Να σημειωθεί επίσης πως η παραγωγή απάντησης στα ελληνικά είναι πιο χρονοβόρα από ότι στα αγγλικά λόγω κόστους χαρακτήρων και τόνων.

Η χρήση του μοντέλου είναι περιορισμένη καθώς σε κάθε run γίνεται μόνο ένα OpenAI API Call που έχει όριο 4096 tokens ( $\approx 12.000$  χαρακτήρες με αγγλικά και  $\approx 3700$  χαρακτήρες με ελληνικά). Αν το όριο ξεπεραστεί παράγεται σφάλμα, συνεπώς δεν γίνεται σε μία εκτέλεση να παραχθούν άπειρες ερωτήσεις και απαντήσεις.

Με βάση τα παραπάνω λοιπόν, είναι πιο ξεκάθαρος ο τρόπος που το μοντέλο διαχειρίζεται ερωτήματα και απαντήσεις και πως υπολογίζεται το κόστος από κάθε λέξη λαμβάνοντας υπόψη ακόμη και τη γλωσσα και τα σημεία στίξης.

## Backend C#

Για να είναι συμβατό το μοντέλο με τις απαιτήσεις της εταιρείας, έφτιαξα παρόμοιο project με χρήση γλώσσας C#. Ωστόσο υπήρξε μεγάλη διαφορά στη κλήση του μοντέλου. Ως προς τις αδυναμίες, το μοντέλο παραμένει εξίσου προβληματικό.

Αντί για τη χρήση έτοιμων μεθόδων όπως στη Python η χρήση OpenAI μοντέλου απαιτεί δημιουργία Http Request σε μορφή .json. Εάν το request γίνει αποδεκτό, παράγεται η απάντηση και αποθηκεύεται στη συνομιλία.

```
-----ANSWER GENERATION-----  
  
// First and foremost, we need the conversation history to have a proper answer direction  
// You can't enter a conversation and answer a question if you don't know what was implied before  
3 references  
public async Task<List<Dictionary<string, string>>> generate_answer(List<Dictionary<string, string>> conversation)  
{  
    // OpenAi API key. Created in OpenAi website  
    // Source of the generator (whose OpenAi API will I use)  
    string apiKey = "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";  
  
    // OpenAi API Endpoint  
    string url = "https://api.openai.com/v1/chat/completions";  
  
    // Client used to send Http Requests (in our case our question)  
    HttpClient client = new HttpClient();  
  
    // Authorization which has the API key as a credential  
    client.DefaultRequestHeaders.Add("Authorization", $"Bearer {apiKey}");  
  
    // We create an empty dictionary to upload all existant messages from the conversation  
    List<Dictionary<string, string>> messages = new List<Dictionary<string, string>>();  
  
    // For every message that was already sent in the conversation  
    foreach (Dictionary<string, string> dict in conversation)  
    {  
        // Get the role from the message (who sent it)  
        string role = dict.TryGetValue("role", out var roleValue) ? roleValue : "user";  
  
        // Get the message content (what was sent)  
        string messagecontent = dict.TryGetValue("content", out var contentValues) ? contentValues : string.Empty;  
  
        // Add the message to the empty list  
        messages.Add(new Dictionary<string, string>()  
        {  
            { "role", role },  
            { "content", messagecontent }  
        });  
    }  
  
    // Data to modify the OpenAi API generator  
    var payload = new  
    {  
        messages, // All sent messages  
        model = "gpt-3.5-turbo", // Which OpenAi API model is used  
        temperature = 0.0 // Produce more deterministic output.  
    };  
    var requestData = JsonConvert.SerializeObject(payload); // convert data to JSON  
  
    // Http request to the API, using JSON data as the request body  
    var content = new StringContent(requestData, Encoding.UTF8, "application/json");  
  
    // We sent the Http request to the https://api.openai.com/v1/chat/completions  
    // so that we get the generated answer  
    // In other words: We ask ChatGpt to produce the answer to our request  
    var response = await client.PostAsync(url, content);  
  
    dynamic responseData = null; // the response from the OpenAi API
```

```

if (response.IsSuccessStatusCode) // If the request is successfully transferred
{
    // We will receive an answer from the API
    string responseJson = await response.Content.ReadAsStringAsync();
    responseData = JsonConvert.DeserializeObject(responseJson); // we get the answer in JSON

    if (responseData != null)
    {
        // Add the new answer to the conversation
        foreach (var choice in responseData.choices)
        {
            var newMessage = new Dictionary<string, string>
            {
                { "role", choice.message.role.ToString() },
                { "content", choice.message.content.ToString() }
            };
            conversation.Add(newMessage);
        }
    }
}

```

Μία ακόμη αλλαγή είναι η προσθήκη ρόλου στο μοντέλο. Εάν στο μοντέλο ως πρώτο prompt θέσουμε “You are ...” το μοντέλο απαντάει με “I am...” και έχει καλύτερη κατεύθυνση για να δίνει απαντήσεις.

```

if (language.Equals("en")) // If we have chosen "en" from the language dropdown list
{
    conversation.Add(new Dictionary<string, string>
    {
        { "role", "user" },
        { "content", "You are an AI assistant." }
    });

    conversation.Add(new Dictionary<string, string>
    {
        { "role", "assistant" },
        { "content", "I am an AI assistant. I will give you answers without any of my comments" }
    });
}
else if (language.Equals("el")) // If we have chosen "el" from the language dropdown list
{
    conversation.Add(new Dictionary<string, string>
    {
        { "role", "user" },
        { "content", "Είσαι ένας AI βοηθός" }
    });

    conversation.Add(new Dictionary<string, string>
    {
        { "role", "assistant" },
        { "content", "Είμαι ένας βοηθός AI. Δίνω απαντήσεις χωρίς δικά μου σχόλια" }
    });
}

```

## Full Stack ASP NET 6.0 Framework

Αφού τροποποίησαμε τη χρήση του μοντέλου GPT, ήρθε η ώρα να το αξιοποιήσουμε σε Web Application, το Eventora S.I.P. Το αντίστοιχο project κατασκευάστηκε ως ASP NET Framework.

Για το frontend μέρος των σελίδων χρησιμοποιούμε αρχεία όψεων (Views) που βρίσκονται σε ξεχωριστό φάκελο του project γραμμένα σε HTML, CSS και JavaScript.

Για το backend μέρος, η ιστοσελίδα έχει έναν Controller, που βρίσκεται επίσης σε ξεχωριστό φάκελο. Οι λειτουργίες του Controller αναπαριστώνται ως ActionResult μέθοδοι οι οποίες συνδυάζονται λειτουργικά με scripts γραμμένα σε JavaScript, τα οποία είναι γραμμένα στα Views, καθώς και με κλάσεις που βρίσκονται στο φάκελο Models.

Για το debugging πολλή χρήσιμη ήταν η χρήση breakpoints που ελέγχουν τη ροή εκτέλεσης του κώδικα ανά εντολή και τις τιμές όλων των αντικειμένων, οπότε μπορούσα να δω όλο το ιστορικό του μοντέλου και να βρω πολύ πιο εύκολα τυχόν ατέλειες και λάθη στον κώδικα. Εξίσου χρήσιμη ήταν η λειτουργία inspect στην ιστοσελίδα, που εμφάνιζε και αυτή σφάλματα.

Αφού έφτιαξα ένα πρωτότυπο του Eventora S.I.P., ο Head Software Developer μου ζήτησε την κατασκευή ενός namespace, με όσα περισσότερα σχόλια γινόταν, που θα έχει κομμάτια κώδικα που αφορούν το μοντέλο GPT. Πέρα από ευκολία ανάγνωσης και κατανόησης, το namespace είναι απαραίτητο καθώς η εταιρεία μπορεί να θέλει να το χρησιμοποιήσει στο μέλλον με διαφορετικό τρόπο και να κάνει αλλαγές στον κώδικα.

### Screenshots από το Eventora S.I.P:

The screenshot shows the Eventora S.I.P. invitation production interface. At the top, there's a header bar with the logo 'ev' and the text 'S.I.P. Invitation Production'. Below it is a navigation bar with icons for back, forward, and search, followed by the URL 'localhost:44301/Eventora\_Smart\_Invitation\_Productor/Production'. The main content area has a dark background with white text and input fields. It includes sections for choosing language ('el'), date ('Start of Event: 18-10-2023', 'End of Event: 18-10-2023'), and time ('Choose starting hour: 09 00', 'Choose ending hour: 14 00'). There are also fields for event category ('Sport') and name ('1ος Αγώνας Τρεξίματος ORAMA'). A large text area at the bottom asks for event info and contains the text: 'Give us some info about the event' and 'Οργανωμένος από την ομάδα τρεξίματος ORAMA. Στην Αγία Παρασκευή, Αθήνα. 7 ευρώ εγγραφή online.' A green button at the bottom right says 'Generate Invitation'.

**Generated answer:** Αγαπητοί φίλοι του αθλητισμού.

Σας προσκαλούμε να συμμετάσχετε στον 1ο Αγώνα Τρεξίματος ORAMA, μια εκδήλωση αθλητικού χαρακτήρα που διοργανώνεται από την ομάδα τρεξίματος ORAMA. Ο αγώνας θα πραγματοποιηθεί στην περιοχή της Αγίας Παρασκευής, Αθήνα, στις 18 Οκτωβρίου 2023, από τις 09:00 έως τις 14:00.

Η συμμετοχή σας μπορεί να γίνει εύκολα και γρήγορα μέσω της ηλεκτρονικής εγγραφής, με κόστος 7 ευρώ. Παρακαλούμε επισκεφθείτε την ιστοσελίδα μας για να ολοκληρώσετε την εγγραφή σας.

Ανυπομονούμε να σας υποδεχθούμε στον 1ο Αγώνα Τρεξίματος ORAMA και να μοιραστούμε μαζί σας μια μοναδική εμπειρία αθλητισμού και διασκέδασης.

Με εκτίμηση,  
Η ομάδα τρεξίματος ORAMA

More friendly   More efficient   More words   Fewer words

Regenerate invitation

Generate registration message

Save Invitation

**✓ Successful registration:** Αγαπητέ/ή [Όνομα συμμετέχοντα].

Σας ευχαριστούμε που εγγραφήκατε με επιτυχία στον 1ο Αγώνα Τρεξίματος ORAMA! Είμαστε πολύ χαρούμενοι που θα σας έχουμε μαζί μας σε αυτήν την αθλητική εκδήλωση.

Η εγγραφή σας έχει καταχωριθεί με επιτυχία και η θέση σας είναι επιβεβαιωμένη. Ανυπομονούμε να σας υποδεχθούμε στην Αγία Παρασκευή, Αθήνα, στις 18 Οκτωβρίου 2023, από τις 09:00 έως τις 14:00.

Μην ξεχάστε να φορέστε τα κατάλληλα αθλητικά ρούχα και να φέρετε μαζί σας την ενέργεια και τον ενθουσιασμό σας. Είμαστε βέβαιοι ότι ο 1ος Αγώνας Τρεξίματος ORAMA θα είναι μια αξέχαστη εμπειρία για όλους.

Αν έχετε οποιεδήποτε ερωτήσεις ή ανάγκη για περαιτέρω πληροφορίες, μην διστάστε να επικοινωνήσετε μαζί μας. Θα είμαστε εδώ για να σας βοηθήσουμε.

Ανυπομονούμε να σας δούμε στον 1ο Αγώνα Τρεξίματος ORAMA και να μοιραστούμε μαζί σας μια μοναδική αθλητική εμπειρία!

Με εκτίμηση,  
Η ομάδα τρεξίματος ORAMA

Save Invitation

**✗ Failed registration:** Αγαπητέ/ή [Όνομα συμμετέχοντα].

Λυπούμαστε πολύ για την ενδιαφέροντας σας και την προσπάθεια που καταβάλλετε για να εγγραφείτε. Λυπούμαστε που δεν μπορέσαμε να σας προσφέρουμε μια θέση σε αυτήν την εκδήλωση.

Σας ευχαριστούμε πολύ για το ενδιαφέρον σας και την προσπάθεια που καταβάλλετε για να εγγραφείτε. Λυπούμαστε που δεν μπορέσαμε να σας προσφέρουμε μια θέση σε αυτήν την εκδήλωση.

Σας ευθαρρύνουμε να παρακολουθήσετε τις μελλοντικές εκδηλώσεις μας και να δοκιμάσετε ξανά την εγγραφή σας. Θα είμαστε πάντα εδώ για να σας υποδεχθούμε και να μοιραστούμε μαζί σας την αθλητική μας πάθηση.

Αν έχετε οποιεδήποτε ερωτήσεις ή ανάγκη για περαιτέρω πληροφορίες, μην διστάστε να επικοινωνήσετε μαζί μας. Θα είμαστε εδώ για να σας βοηθήσουμε.

Σας ευχόμαστε καλή συνέχεια και ελπίζουμε να σας δούμε σε μελλοντικές εκδηλώσεις μας.

Με εκτίμηση,  
Η ομάδα τρεξίματος ORAMA

Οι είσοδοι που συμπληρώσαμε αποθηκεύονται ως δεδομένα σε ένα AJAX post μέσω jquery το οποίο αποστέλλεται σε έναν server που δρομολογεί τα δεδομένα στην κατάλληλη μέθοδο ως παραμέτρους. Με βάση τη γλώσσα επιλέγεται η λίστα prompts που θα αξιοποιήσουμε και δίνουμε ρόλο στο μοντέλο μέσω “You are...” ή “Είσαι...”.

Αφού τελειώσει η παραγωγή θα πρέπει να φαίνονται κάποιες από τις εισόδους που ήδη συμπληρώσαμε και ακριβώς από κάτω το κείμενο της πρόσκλησης.

Πριν επιλέξει φίλτρο αναπαραγωγής, ο χρήστης μπορεί να διορθώσει κάποια από τα πεδία που συμπλήρωσε προηγουμένως για να αποφύγει ανασύνταξη όσων έγραψε επιστρέφοντας στην αρχική σελίδα.

Τέλος, το πρόβλημα των tokens από τις προηγούμενες φάσεις λύθηκε. Η αποστολή των prompts στο μοντέλο γίνεται με κάθε κλήση ActionResult, που το καθένα πραγματοποίει ένα νέο OpenAI Api call.

## Τι πετύχαμε

- Παραγωγή σε μόνο λίγα λεπτά ή και δευτερόλεπτα
- Απαλλαγή από ορθογραφικά λάθη
- Ευκολία χρήσης με συγγραφή κειμένου, επιλογή από μενού και πάτημα κουμπιών
- Δυνατότητα άμεσης τροποποίησης και αποθήκευσης μέσω φίλτρων ή επεξεργασίας του .txt από τον ίδιο τον χρήστη
- Εξοικονόμηση χρόνου
- Εύκολη διόρθωση και τροποποίηση του κώδικα του μοντέλου μέσω του namespace

## Eventora Spread The Word

### Εισαγωγή

Σε κάποιες εκδηλώσεις, παρέχονται ερωτηματολόγια ή φόρμες όπου οι συμμετέχοντες ζητούνται να διατυπώσουν σχόλια. Οι διοργανωτές διαβάζοντας τα μπορούν να δουν τι εντυπώσεις δημιουργήθηκαν, να διακρίνουν τα στοιχεία που άρεσαν ή δεν άρεσαν στον κόσμο, ούτως ώστε να γνωρίζουν τις δυνάμεις και αδυναμίες της εκδήλωσης τους.

Προφανώς η καταμέτρηση, η ανάγνωση και η αξιολόγηση σχολίων είναι πολύ χρονοβόρες, ειδικά αν υπάρχουν πολλά σχόλια, ακόμη και αν μοιραστεί η δουλειά σε πολλά άτομα.

Η λύση σε αυτό το πρόβλημα είναι το **Eventora Spread The Word**. Στόχος του είναι για κάθε εκδήλωση μέσω Μηχανικής Μάθησης να συλλέγει, να αξιολογεί τα σχόλια της ως “Positive”, “Negative” ή και “Neutral” και στο τέλος τα προβάλει μαζί με διαγράμματα μία γενική αξιολόγηση της εκδήλωσης.

### Long Short Term Memory

Το μοντέλο Μηχανικής Μάθησης που χρησιμοποίησα αποκαλείται Long Short Term Memory (LSTM). Είναι ένα είδος νευρωνικού δικτύου που χρησιμοποιείται για να κατανοήσει και να αντιμετωπίσει μακροπρόθεσμες σχέσεις στα δεδομένα, όπως χρονοσειρές ή φράσεις σε φυσική γλώσσα. Βασικό χαρακτηριστικό του είναι οι

πύλες που ελέγχουν τη ροή των πληροφοριών μέσα και έξω από κάθε κυψέλη του δικτύου, επιτρέποντας την αποτελεσματική διαχείριση της πληροφορίας.

## Sentiment Analysis

Η ανάλυση συναισθημάτων, Sentiment Analysis είναι μια τεχνική που χρησιμοποιείται για τον προσδιορισμό και αξιολόγηση του συναισθηματικού τόνου που περιέχεται σε κείμενα, όπως κριτικές, σχόλια και άλλα. Χρησιμοποιείται ευρέως σε εφαρμογές όπως παρακολούθηση της κοινής γνώμης, αξιολόγηση και παρακολούθηση της αντίδρασης του κοινού σε κοινωνικά μέσα.

## Κατασκευή μοντέλου μέσω Google Colab

Εκπαίδευσα ένα μοντέλο LSTM το οποίο δέχεται πολλαπλά σχόλια και επιστρέφει τις κατάλληλες αξιολογήσεις. Για τη κατασκευή του μοντέλου χρησιμοποίησα notebook στο Google Colab. Το μοντέλο αποθηκεύεται σε κατάλληλη μορφή αρχείου για να μπορεί να χρησιμοποιηθεί οποιαδήποτε στιγμή. Για την εκπαίδευση χρειάζεται:

- Μία βάση δεδομένων σε αρχείο .csv 50.000 σχόλια IMDB που αφορούν ταινίες
- Ένα αρχείο .json για τη μετατροπή λέξεων σε tokens και
- Ένα αρχείο .txt για τη μετατροπή των tokens σε ακολουθίες διανυσμάτων(sequences)

Αρχικά τα δεδομένα χωρίζονται για εκπαίδευση (80%) και δοκιμή (20%). Πριν την εκπαίδευση μετατρέπουμε τα δεδομένα σε κατάλληλα διανύσματα μήκους 100 και προετοιμάζουμε το λεξιλόγιο του μοντέλου για να δοθούν αριθμητικές τιμές στις λέξεις.

```
✓ 0s [23] X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=42)

✓ 7s [24] word_tokenizer = Tokenizer()
         word_tokenizer.fit_on_texts(X_train)

         X_train = word_tokenizer.texts_to_sequences(X_train)
         X_test = word_tokenizer.texts_to_sequences(X_test)

✓ 1s [25] tokenizer_json = word_tokenizer.to_json()
         with io.open('/content/drive/MyDrive/Colab Data/b3_tokenizer.json', 'w', encoding='utf-8') as f:
             f.write(json.dumps(tokenizer_json, ensure_ascii=False))

✓ 0s [26] vocab_length = len(word_tokenizer.word_index) + 1
         vocab_length

92394
```

```

✓ 0s [27] maxlen = 100
      X_train = pad_sequences(X_train, padding='post', maxlen=maxlen)
      X_test = pad_sequences(X_test, padding='post', maxlen=maxlen)

✓ 12s [28] embeddings_dictionary = dict()
      glove_file = open('/content/drive/MyDrive/Colab Data/a2_glove.6B.100d.txt', encoding="utf8")

      for line in glove_file:
          records = line.split()
          word = records[0]
          vector_dimensions = asarray(records[1:], dtype='float32')
          embeddings_dictionary [word] = vector_dimensions
      glove_file.close()

✓ 0s [29] embedding_matrix = zeros((vocab_length, 100))
      for word, index in word_tokenizer.word_index.items():
          embedding_vector = embeddings_dictionary.get(word)
          if embedding_vector is not None:
              embedding_matrix[index] = embedding_vector

```

```

✓ 0s [30] embedding_matrix.shape
      (92394, 100)

✓ 1s [31] lstm_model = Sequential()
      embedding_layer = Embedding(vocab_length, 100, weights=[embedding_matrix], input_length=maxlen , trainable=False)

      lstm_model.add(embedding_layer)
      lstm_model.add(LSTM(128))

      lstm_model.add(Dense(1, activation='sigmoid'))

```

Η εκπαίδευση και ο έλεγχος της ακρίβειας πραγματοποιείται σε 6 κομμάτια. Στο τέλος αποθηκεύουμε το αντικείμενο του μοντέλου σε αρχείο keras .h5

```

✓ 9m [33] lstm_model_history = lstm_model.fit(X_train, y_train, batch_size=128, epochs=6, verbose=1, validation_split=0.2)
      Epoch 1/6
      250/250 [=====] - 87s 336ms/step - loss: 0.5152 - acc: 0.7429 - val_loss: 0.4363 - val_acc: 0.8044
      Epoch 2/6
      250/250 [=====] - 105s 420ms/step - loss: 0.4137 - acc: 0.8133 - val_loss: 0.4054 - val_acc: 0.8301
      Epoch 3/6
      250/250 [=====] - 96s 385ms/step - loss: 0.3724 - acc: 0.8376 - val_loss: 0.3638 - val_acc: 0.8464
      Epoch 4/6
      250/250 [=====] - 86s 343ms/step - loss: 0.3443 - acc: 0.8502 - val_loss: 0.3348 - val_acc: 0.8589
      Epoch 5/6
      250/250 [=====] - 90s 361ms/step - loss: 0.3193 - acc: 0.8661 - val_loss: 0.3265 - val_acc: 0.8615
      Epoch 6/6
      250/250 [=====] - 89s 357ms/step - loss: 0.3039 - acc: 0.8729 - val_loss: 0.3282 - val_acc: 0.8618

✓ 41s [34] score = lstm_model.evaluate(X_test, y_test, verbose=1)
      313/313 [=====] - 28s 90ms/step - loss: 0.3353 - acc: 0.8619

```

Μετά από πολλές δοκιμές, η καλύτερη ακρίβεια σε μοντέλο έφτανε το 86.7%. Στα τελικά cells του notebook βάλαμε και κάποια παραδείγματα.

Σε αυτό το σημείο πρέπει να τονίσουμε πως το μοντέλο δεν δέχεται άμεσα τα strings ως εισόδους, ούτε παράγει εξόδους “Positive”, “Negative”, “Neutral”. Σαν είσοδο πρέπει να δεχτεί προ επεξεργασμένα sequences και σαν έξοδο να παράγει έναν πίνακα με float τιμές.

```

1s [42] with open('/content/drive/MyDrive/Colab Data/b3_tokenizer.json') as f:
    data = json.load(f)
    loaded_tokenizer = tokenizer_from_json(data)

✓ [43] loaded_tokenizer

✓ [44] unseen_tokenized = loaded_tokenizer.texts_to_sequences(reviews)

✓ [45] unseen_tokenized
    ↳ unseen_tokenized

✓ [46] unseen_padded = pad_sequences(unseen_tokenized, padding='post', maxlen=100)

✓ [47] unseen_padded
    ↳ array([[ 304, 1426,   10, ...,     0,     0,     0],
           [11889,     0,     0, ...,     0,     0,     0],
           [ 300,      5, 181, ...,     0,     0,     0],
           ...,
           [ 66,      0,     0, ...,     0,     0,     0],
           [30194,      0,     0, ...,     0,     0,     0],
           [ 486,      0,     0, ...,     0,     0,     0]], dtype=int32)

✓ [48] unseen_sentiments = pretrained_lstm_model.predict(unseen_padded)

✓ [49] unseen_sentiments
    ↳ [array([ 0.6540559], dtype=float32),
        array([ 0.74253416], dtype=float32),
        array([ 0.5303917]), ...]

✓ [50] sentiment_categories = []
      for sentiment_score in final_results:
          if sentiment_score > 8.49:
              sentiment_categories.append("Positive")
          elif sentiment_score >= 5.0:
              sentiment_categories.append("Neutral")
          else:
              sentiment_categories.append("Negative")

```

## Χρήση του μοντέλου σε Python

Η χρήση του έτοιμου μοντέλου μηχανικής μάθησης δεν αποτελεί καμία δοκιμασία. Φέρνουμε τα strings σε κατάλληλη μορφή, να φορτώσουμε μέσω `.load()` το αρχείο του μοντέλου και να καλέσουμε την μέθοδο `.predict()`.

```

28
29     array_of_strings_test = ["neutral", "positive", "negative"]
30     json_for_comments_test = json.dumps(array_of_strings_test)
31     mp_test = 'lstm/model.h5'
32     tp_test = 'lstm/tokenizer.json'
33     results = predict(mp_test, tp_test, json_for_comments_test)

Run main ×
Run | :
2023-10-02 11:20:28.804100: I tensorflow/core/platform/cpu_feature_guard.cc:182] This TensorFlow
To enable the following instructions: SSE SSE2 SSE3 SSE4.1 SSE4.2, in other operations, rebuild T
1/1 [=====] - 2s 2s/step
Output: [[0.6540559]
          [0.74253416]
          [0.5303917]]

```

## Χρήση του μοντέλου σε C#

Σε σύγκριση με τη Python, η υλοποίηση του μοντέλου είναι αρκετά πιο δύσκολη σε C# για τους εξής λόγους:

- Δεν στηρίζονται οι βιβλιοθήκες tensorflow, keras
- Τα μοντέλα μηχανικής μάθησης σε .h5 δεν στηρίζονται
- Οι μέθοδοι για μετατροπή strings σε sequences δεν υπάρχουν
- Το αντικείμενο Tokenizer της Python δεν υπάρχει

Η λύση στα παραπάνω προβλήματα είναι να φέρουμε το μοντέλο σε μια μορφή αποδεκτή από το Visual Studio Code σε C# και όσες μέθοδοι αξιοποιούν οι βιβλιοθήκες να τις κατασκευάσουμε εμείς οι ίδιοι ούτως ώστε να εμφανίζονται ίδια αποτελέσματα για τις ίδιες εισόδους.

```
public class Tokenizer
{
    /*
     * The Tokenizer class is used transform all strings' words to integer values and the
     * creates padding sequences which are used as input for the LSTM model
     */
    public Preprocessor preprocessor = new Preprocessor(); // Preprocessor for text

    public Natural_Language_Toolkit nltk = new Natural_Language_Toolkit(); // Toolkit to load stopwords
    0 references
    public string class_name { get; set; } // JSON class_name property
    3 references
    public Config config { get; set; } // JSON config property
    0 references
    public Tokenizer() // default constructor
    {
        config = new Config(); // Set Tokenizer Configuration
    }
}
```

```
2 references
public int[] TokenizeSentence(string sentence, Dictionary<string, int> wordIndex) // create word tokens
{
    List<string> stop_words = nltk.GetStopWords(); // Get stopwords

    sentence = preprocessor.PreprocessText(sentence, stop_words); // Preprocess text

    string[] words = sentence.Split(' '); // Separate sentence's words

    List<int> tokenizedSequence = new List<int>(); // List of int word values

    foreach (string word in words) // For every word
    {
        if (wordIndex.TryGetValue(word, out int index)) // Search in the dictionary
        {
            tokenizedSequence.Add(index);
        }
    }
    return tokenizedSequence.ToArray(); // Get int array
}
```

```
2 references
public List<int[]> PadSequences(List<int[]> sequences, int maxLength) // Pad int sequences
{
    List<int[]> paddedSequences = new List<int[]>(); // prepare the sequences

    foreach (var sequence in sequences)
    {
        int sequenceLength = Math.Min(sequence.Length, maxLength); // the max length for all sequences is 100

        int[] paddedSequence = new int[maxLength]; // we have a int[100] array

        Array.Copy(sequence, paddedSequence, sequenceLength); // we give all int values we can find

        for (int i = sequenceLength; i < maxLength; i++)
        {
            paddedSequence[i] = 0; // if the sentence does not cover all 100 spaces we set the rest to 0
        }

        paddedSequences.Add(paddedSequence);
    }
    return paddedSequences;
}
```

```

        var tensor = new DenseTensor<float>(inputData.ToArray(), inputShape);

        string inputName = onnxModel.InputMetadata.Keys.First();

        var inputs = new List<NamedOnnxValue> { NamedOnnxValue.CreateFromTensor(inputName, tensor) };
        var results = onnxModel.Run(inputs);

        var outputName9 = onnxModel.OutputMetadata.Keys.First();
        var outputTensor = results.FirstOrDefault(output => output.Name == outputName9)?.AsTensor<float>();

        var predictions = outputTensor.ToArray();

        return predictions;
    }
    else
    {
        return null;
    }
}
catch (Exception ex)
{
    Console.WriteLine($"Error predicting sentiment: {ex.Message}");
    return null;
}

```

Ωστόσο, ο κώδικας C# που κατασκευάσαμε για το μοντέλο μας μπορεί να χρησιμοποιηθεί μόνο εκπαιδευτικά για να εμβαθύνουμε στο τι ακριβώς κάνουν οι ισοδύναμες μέθοδοι Python και για πολύ λίγα δεδομένα. Παρακάτω παρουσιάζονται κάποια από τα αποτελέσματα δοκιμών. Οι διαφορές είναι δραματικά μεγάλες και κάνουν ξεκάθαρο πως η χρήση του μοντέλου πρέπει να γίνει με Python.

Σενάρια	Αποτέλεσμα	Χρόνος αναμονής	Ταχύτητα
Python 100 σχόλια	float32 array	0.5- δευτερόλεπτα	Στιγμιαία
C# 100 σχόλια	float[] array	3- λεπτά	Πολύ αργή
Python 1000 σχόλια	float32 array	3- δευτερόλεπτα	Σχεδόν στιγμιαία
C# 1000 σχόλια	runtime exception	15+ λεπτά	Μη-αξιόπιστη
Python 4000 σχόλια	float32 array	12- δευτερόλεπτα	Γρήγορη
C# 4000 σχόλια	runtime exception	15+ λεπτά	Μη-αξιόπιστη

## Full Stack ASP NET 6.0 Framework

Παρόλο που υπάρχει ακόμα το πρόβλημα ταχύτητας με τη χρήση των δικών μας μεθόδων, θα επιδιώξουμε να κατασκευάσουμε ένα πρωτότυπο web application για μελλοντική τροποποίηση. Για ευκολία πρόσβασης και δόμησης, οι εκδηλώσεις και τα σχόλια που βρίσκονται σε ένα απλό .xml αρχείο. Στόχος μας είναι να αναπτύξουμε κώδικα χρήσης του μοντέλου μέσω C# και στο μέλλον να δούμε πως μπορούμε να τον αντικαταστήσουμε με μία πιο γρήγορη λύση.

## Screenshots από το Eventora S.T.W.:

The screenshot shows the 'Event List' section of the Eventora S.T.W. interface. At the top, there's a navigation bar with the 'eventora' logo and two links: '1. Smart Invitation Producer' and '2. Spread The Word'. Below the navigation is a title 'Event List' and a sub-instruction 'Choose an event from the list below.'. A 'Toggle Language' button is also present. The main area features a grid of event cards. One card is highlighted with a green border and contains a silhouette of people. To the right of this card, the text 'Test Rank' is displayed. Below the card, the text 'Last Visited: 2023-10-02' and 'Rank: Good' is shown. A red arrow points from the text 'Rank: Good' towards the 'Test Rank' label.

The screenshot shows the 'Event Details' section. The top navigation bar includes the 'eventora' logo and the same two links: '1. Smart Invitation Producer' and '2. Spread The Word'. Below the navigation, there are three tabs: 'Positive Comments' (selected), 'Negative Comments', and 'Event's Rank'. The main content area displays a large, partially visible image of a group of people. Below the image, the title 'Test Rank' is shown, followed by the update date 'Update: 2023-10-02' and the rank 'Rank: Ok'. The section is titled 'Comments: 12'.

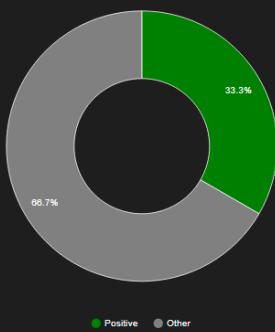
This screenshot shows the detailed view of the 'Test Rank' section. It starts with the title 'Test Rank' and the update date 'Update: 2023-10-02'. Below that is the rank 'Rank: Ok'. The section is titled 'Comments: 12'. A numbered list follows, detailing 12 comments with their scores:

1. Ήταν μία πολύ ευχάριστη εμπειρία Score: 0,95 Positive
2. Ήταν μία απαίσια εμπειρία Score: 0,13 Negative
3. Δεν θα ξαναέρθω ποτέ Score: 0,69 Neutral
4. Ουδέτερος Score: 0,65 Neutral
5. Ήταν καλό Score: 0,77 Positive
6. Τίποτα ιδιαιτέρω Score: 0,67 Neutral
7. Χάλια Score: 0,30 Negative
8. Δεν μου άρεσ Score: 0,65 Neutral
9. Δεν μου άρεσε καθόλου Score: 0,65 Neutral
10. ΤΕΛΕΙΟ Score: 0,90 Positive
11. Την επόμενη φορά θα φέρω και τον φίλο μου μαζί Score: 0,72 Positive
12. Ουδέν σχόλιο Score: 0,67 Neutral

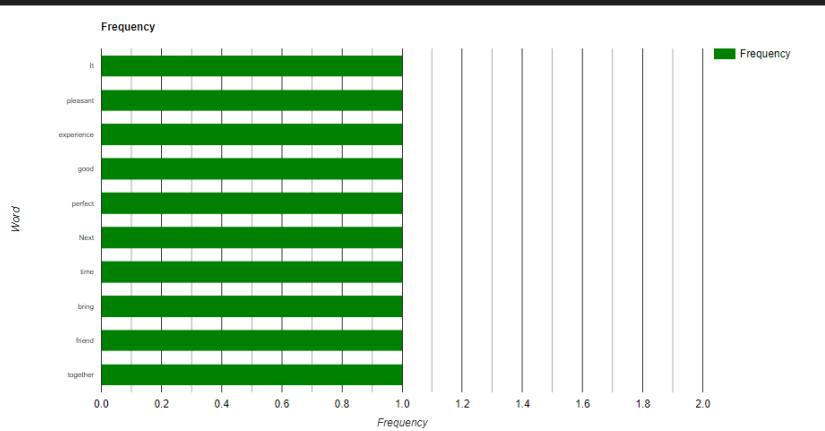
## Positive Comments: 4

1. Ήταν μία πολύ ευχάριστη εμπειρία
2. Ήταν καλό
3. ΤΕΛΙΟ
4. Την επόμενη φορά θα φέρω και τον φίλο μου μαζί

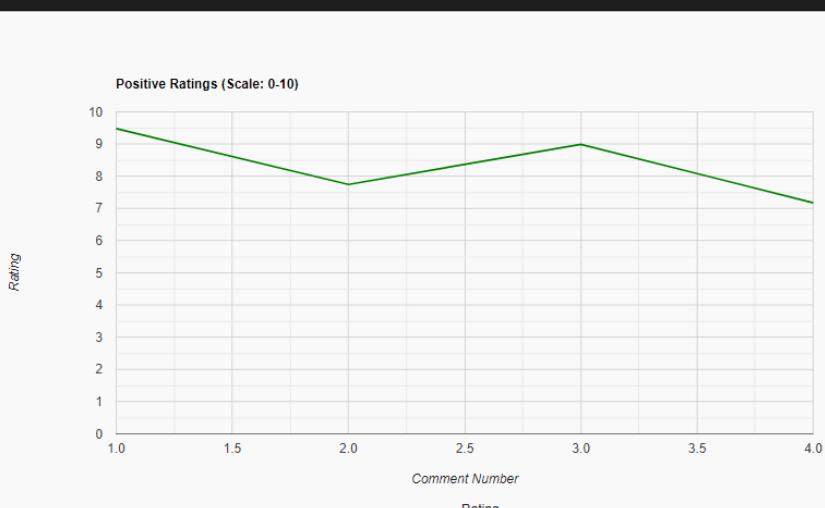
## Positive Sentiments Donut Chart

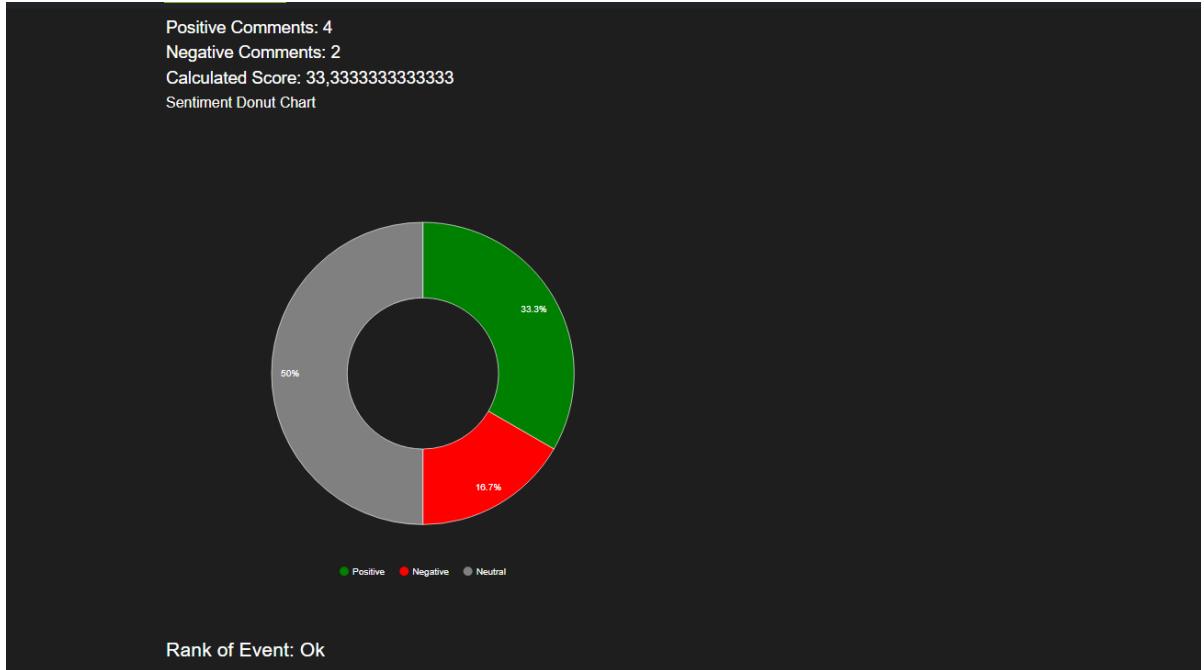


## Word Frequency Chart



## Ratings Line Chart





## Χρήση του μοντέλου για ελληνικά σχόλια

Η βάση δεδομένων στην οποία εκπαιδεύτηκε το μοντέλο μας ήταν 50.000 κριτικές ταινιών στα αγγλικά. Αυτό δημιουργεί πρόβλημα στη δική μας περίπτωση καθώς έχουμε να κάνουμε και με κριτικές στα ελληνικά. Μιάς και έχουμε ήδη το μοντέλο έτοιμο, μία λύση στο πρόβλημα είναι όλα τα ελληνικά σχόλια να τα μεταφράσουμε στα αγγλικά κατά τη διάρκεια της προεπεξεργασίας.

Στη C#, υπάρχουν μεταφραστικά API υπό χρέωση, ωστόσο διαλέξαμε μία προσωρινή λύση. Στην Python παρέχεται η βιβλιοθήκη googletrans η οποία μπορεί να μεταφράσει οποιοδήποτε κείμενο σε ό,τι γλώσσα επιθυμούμε.

```

File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help | Search | Event
Debug Any CPU IIS Express (Google Chrome) | 
AWS Explorer OpenAi_Generator.cs Events.xml Eventora_Smart...rController.cs translation.py Eventora_SpreadController.cs
from googletrans import Translator
import sys

def translate_to_english(text):
    translator = Translator()
    translated_text = translator.translate(text, src='auto', dest='en').text
    return translated_text

if __name__ == "__main__":
    input_text = sys.argv[1]
    english_text = translate_to_english(input_text)
    print(english_text)

```

Καθώς το project είναι σε C# είναι αδύνατο έτσι απλά να εκτελεστεί κώδικας Python. Αφού έφτιαξα ένα .py αρχείο μετάφρασης, έκανα τα εξής βήματα:

- Εγκατάσταση της βιβλιοθήκης Python στο NuGet Package Manager του project
- Εγκατάσταση της βιβλιοθήκης googletrans μέσω pip install στη συσκευή
- Αποθήκευση του .py αρχείου μετάφρασης στα αρχεία του project
- Προσθήκη path για το αρχείο python.exe στις environmental variables της συσκευής
- Εκτελούμε python script με είσοδο το string που θέλουμε να μεταφράσουμε

Παρόλο που το πρόβλημα των ελληνικών λύθηκε δημιουργήθηκαν κάποια νέα θέματα.

Πρώτα από όλα η εκτέλεση του script εξαρτάται από τη συσκευή του χρήστη και τις environmental variables του. Επίσης, για να τρέξει το project ο χρήστης πρέπει να κατεβάσει την βιβλιοθήκη googletrans.

```
1 reference
private static string FindPythonPath() // Method to find the python.exe executable file
{
    string[] paths = Environment.GetEnvironmentVariable("PATH").Split(';'); // find all environmental
    foreach (string path in paths) // Search for the executable
    {
        string pythonPath = Path.Combine(path, "python.exe");

        // Avoid Microsoft of WindowsApps store python.exe
        // (Possibility to collide with device's already existent Python executable)
        if (File.Exists(pythonPath) && !path.Contains("Microsoft") && !path.Contains("WindowsApps"))
        {
            return pythonPath;
        }
    }

    return null;
}
```

```
1 reference
public string TranslateToEnglish(string inputText)
{
    //-----EXECUTE PYTHON SCRIPT-----
    string pythonPath = FindPythonPath(); // Find python.exe executable from the environment variables
    /* !Note: If you wish to execute the program
     * 1) Make sure you have downloaded Python in your device
     * 2) Preferably use 3.11 edition
     * 3) Add the python.exe location to your environmental variables
     */
    string scriptPath = Path.Combine(AppDomain.CurrentDomain.BaseDirectory, "Python", "translation.py");
                                            // It uses googletrans Translator library

    ProcessStartInfo psi = new ProcessStartInfo // How the script will be executed
    {
        FileName = pythonPath,
        Arguments = $"\"{scriptPath}\\" \"{inputText}\"",
        RedirectStandardOutput = true,
        UseShellExecute = false,
        CreateNoWindow = true
    };

    using (Process process = new Process()) // Begin .py file execution
    {
        process.StartInfo = psi;
        process.Start();

        string output = process.StandardOutput.ReadToEnd(); // The English text of interest
        process.WaitForExit();

        return output.Trim();
    }
}
```

## **Amazon Web Services (AWS)**

Στο πρωτότυπο project όπου το μοντέλο μηχανικής μάθησης αξιολογεί αγγλικά και ελληνικά σχόλια, μέσω των δικών μου μεθόδων σε C# υπάρχουν τα εξής κρίσιμα σημεία:

1. Η ταχύτητα του μοντέλου παραμένει δραματικά αργή
2. Η μετάφραση των ελληνικών σχολίων σε αγγλικών εξαρτάται από τη συσκευή του χρήστη
3. Η αξιολόγηση γίνεται πάντα από το πρώτο έως το τελευταίο σχόλιο, δεν υπάρχει αποθήκευση ήδη αξιολογούμενων σχολίων
4. Το αρχείο του μοντέλου βρίσκεται μέσα στο project (προτιμότερο θα ήταν ένας δευτερεύοντας αποθηκευτικός χώρος για ασφάλεια)
5. Το web application λειτουργεί, έστω καθυστερημένα, για το πολύ 100 σχόλια

Εάν επιλυθούν αυτά το Eventora S.T.W. θα λειτουργήσει πιο γρήγορα για ακόμη περισσότερα σχόλια, ανεξαρτήτως συσκευής.

Σε αυτό το σημείο, η εταιρεία μου σύστησε μία πλατφόρμα cloud computing που μπορεί να βοηθήσει και την χρησιμοποιεί αρκετά συχνά, την **Amazon Web Services (AWS)**. Παρέχει online εργαλεία που ειδικεύονται στην ανάπτυξη και λειτουργία εφαρμογών και δικών μας υπηρεσιών.

Για να έχω πρόσβαση στη AWS, η εταιρεία μου έφτιαξε λογαριασμό με τις απαραίτητες άδειες και δικαιώματα πρόσβασης για να δουλέψω ελεύθερα.

## **Πρόσβαση δεδομένων μέσω AWS**

Ένα από τα θέματα που έχουμε στο πρωτότυπο web application είναι ο τρόπος με τον οποίο τα αρχεία του μοντέλου είναι αποθηκευμένα. Το AWS παρέχει λύση σε αυτό μέσω των S3 buckets.

```

try
{
    string access_key = "AKIAW7CT25U43NRICWCE";
    string secret_key = "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";
    var credentials = new BasicAWSCredentials(access_key, secret_key);

    Console.WriteLine("User Access key: " + credentials.GetCredentials().AccessKey);
    Console.WriteLine("User Secret key: " + credentials.GetCredentials().SecretKey);
    Console.WriteLine();

    var config = new AmazonS3Config
    {
        RegionEndpoint = Amazon.RegionEndpoint.USEast1
    };

    var s3Client = new AmazonS3Client(credentials, config);

    Console.WriteLine("AmazonS3Client created successfully: " + s3Client.ToString());
    Console.WriteLine();

    var bucketName = "lstmbucket";
    var prefix = "models/model.onnx";
    var prefix3 = "tokenizers/tokenizer.json";

    ListObjectsV2Request request = new ListObjectsV2Request
    {
        BucketName = bucketName,
        Prefix = prefix
    };

    ListObjectsV2Response response = await s3Client.ListObjectsV2Async(request);
}

```

Μέσω αυτών μπορούμε να οργανώσουμε φακέλους και αρχεία και να αποκτήσουμε πρόσβαση σε αυτά μέσω του Visual Studio Code. Η απόκτηση πρόσβασης είναι εύκολη, καθώς χρειάζονται δύο κλειδιά και αποστολή ενός request ως S3 Client.

## Κατασκευή και χρήση μεθόδων μέσω AWS

Ένα άλλο θέμα που αναφέραμε είναι η εξάρτηση του κώδικα μετάφρασης python από τη συσκευή. Το AWS, για ακόμα μία φορά έχει τη λύση, τα Lambda Functions. Τα Lambda Functions είναι μέθοδοι που μπορούν να γραφτούν σε Python, Node JS, C#, Powershell, Ruby, Java (σε συγκεκριμένες εκδόσεις).

```

//-----TEST 2: Python AWS Lambda function-----
if (test_transformation_lambda_function)
{
    string access_key = "AKIAW7CT25U43NRICWCE";
    string secret_key = "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";

    AmazonLambdaClient lambdaClient = new AmazonLambdaClient(access_key, secret_key, Amazon.RegionEndpoint.EUWest3);

    Console.WriteLine("Please enter some text:");
    string inputString = Console.ReadLine();

    InvokeRequest request = new InvokeRequest
    {
        FunctionName = "Translator",
        Payload = $"{{\"myString\": \"{inputString}\", \"language\": \"en-es\"}}",
        InvocationType = InvocationType.RequestResponse
    };

    try
    {
        InvokeResponse response = await lambdaClient.InvokeAsync(request);
        string responseBody = Encoding.UTF8.GetString(response.Payload.ToArray());
        Console.WriteLine("Response: " + responseBody);
        JObject jsonResponse = JObject.Parse(responseBody);
        string translatedText = jsonResponse["body"].ToString();
        Console.WriteLine("Translated Text: " + translatedText);
    }
    catch (Exception ex)
    {
        Console.WriteLine("Error: " + ex.Message);
    }
}

```

To AWS επιτρέπει την αποθήκευση τρίτων βιβλιοθηκών σε Layers τα οποία μετά μπορούν να μπουν στο Lambda function. Το μόνο που χρειάζεται ένα Layer είναι ένα

.zip ενός φακέλου με το όνομα python (αυστηρά αυτό το όνομα) στο οποίο μέσω Powershell εκτελούμε pip install τη βιβλιοθήκη που θέλουμε.

Αφού ετοιμάστηκε το Layer με τη googletrans, το προσθέσαμε στη Lambda Function μετάφραση κειμένου και πλέον μένει η υλοποίηση της στο Eventora S.T.W..

## Sagemaker

Μέχρι τώρα έχει λυθεί το πρόβλημα του Python script και ο τρόπος αποθήκευσης και πρόσβασης αρχείων. Πλέον μένει το πιο σοβαρό πρόβλημα που είναι η προεπεξεργασία των σχολίων και η ταχύτητα του μοντέλου. Το AWS έχει μία ειδική υπηρεσία που αφορά μοντέλα Μηχανικής Μάθησης και μάλιστα παρέχει τρόπο άμεσης χρήσης τους μέσω endpoints. Η υπηρεσία αυτή αποκαλείται SageMaker.

## Κατασκευή μοντέλου μοντέλου Sagemaker

Για τη κατασκευή του μοντέλου θα χρησιμοποιήσουμε το **εργαλείο SageMaker Studio** που ειδικεύεται στην εκπαίδευση και στην αποθήκευση μοντέλων Μηχανικής Μάθησης και χρησιμοποιεί Notebooks, όπως και το Jupiter.



Για να χρησιμοποιηθεί χρειάζεται η κατασκευή ενός περιβάλλοντος (domain) με και η επιλογή κατάλληλου πυρήνα (kernel). Στη περίπτωση μας θα επιλέξουμε kernel με Python 10 και Tensorflow 2.13 GPU.

Η προσέγγιση μας είναι παρόμοια με το Google Colab όμως υπάρχουν οι εξής διαφορές:

- Τα αρχεία .csv, .json, .txt της εκπαίδευσης βρίσκονται σε S3 bucket
- Ο κώδικας εκπαίδευσης είναι σε ξεχωριστό cell και μέσω εντολής %write τον αποθηκεύουμε αρχείο .py (inference.py)

**Screenshots από την κατασκευή κώδικα εκπαίδευσης και χρήσης του μοντέλου**

```

print("-----[TASK] Installing Argument Parser-----")
parser = argparse.ArgumentParser() # The Parser is used when we want to call the script and we have to give specific parameters

if parser is not None:
    print("Argument Parser installed")
else:
    print("Argument Parser is None")
print("-----\n")

print("-----[TASK] Add arguments to Parser-----")
parser.add_argument("--model-dir", type=str, default=os.environ.get("SM_MODEL_DIR")) # directory to access files
print("SageMaker Model directory set")

parser.add_argument("--train", type=str, default=os.environ.get("SM_CHANNEL_TRAIN")) # training data directory
print("SageMaker train channel directory set")

parser.add_argument("--test", type=str, default=os.environ.get("SM_CHANNEL_TEST")) # testing data directory
print("SageMaker test channel directory set")

parser.add_argument("--train-file", type=str, default="train-data.csv") # training data
print("Train data set")

parser.add_argument("--test-file", type=str, default="test-data.csv") # testing data
print("Test data set")
print("-----\n")

cwd = os.getcwd()
print("Current Working Directory:", cwd)

```

```

print("-----[TASK] Load Tokenizer-----")
word_tokenizer = Tokenizer()
print("Fit train data on texts")
word_tokenizer.fit_on_texts(X_train)
print("Train data sequences")
X_train = word_tokenizer.texts_to_sequences(X_train)
print("Test data sequences")
X_test = word_tokenizer.texts_to_sequences(X_test)
print("-----\n")

print("-----[TASK] Write Json Tokenizer-----")
tokenizer_json = word_tokenizer.to_json()
print("Opening tokenizer file")
with io.open(local_tokenizer_md, 'w', encoding='utf-8') as f:
    f.write(json.dumps(tokenizer_json, ensure_ascii=False))
print("-----\n")

```

```

vocab_length = len(word_tokenizer.word_index) + 1
print("[INFO] Vocabulary length: ", vocab_length)
maxlen = 100
print("[INFO] Max length of sequences: ", maxlen)

print("-----[TASK] Pad Sequences-----")
print("Pad train data to sequences")
X_train = pad_sequences(X_train, padding='post', maxlen=maxlen)
print("Pad test data to sequences")
X_test = pad_sequences(X_test, padding='post', maxlen=maxlen)
print("-----\n")

```

```

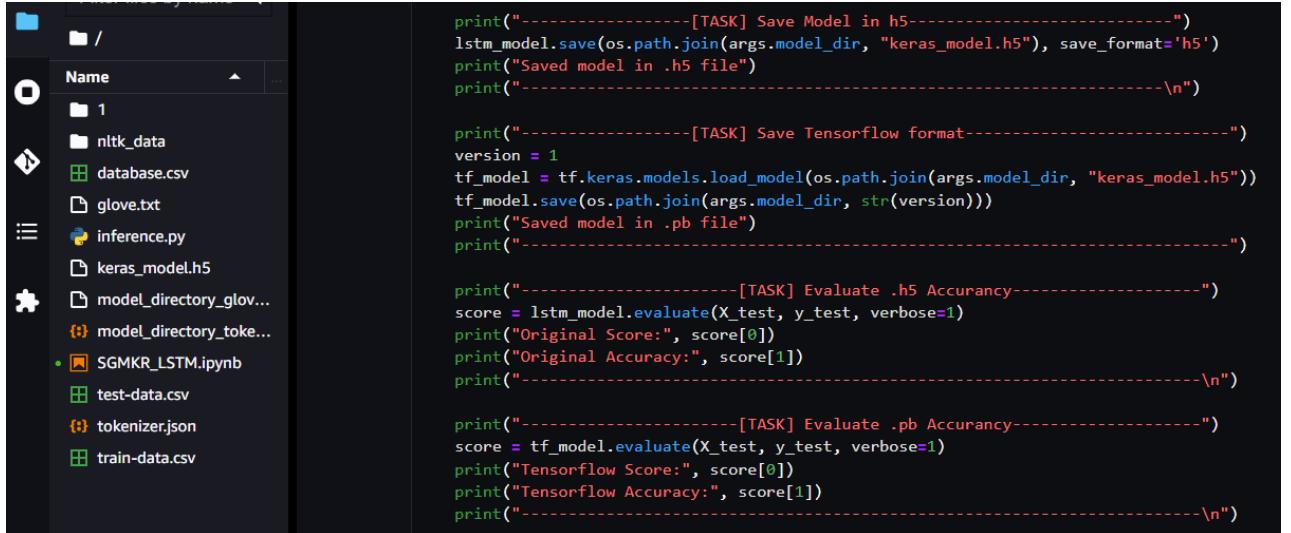
lstm_model = Sequential()
embedding_layer = Embedding(vocab_length, maxlen, weights=[embedding_matrix], input_length=maxlen , trainable=False)

print("Add necessary layers")
lstm_model.add(embedding_layer)
lstm_model.add(LSTM(128))
lstm_model.add(Dense(1, activation='sigmoid'))

print("Compile model")
lstm_model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['acc'])
print(lstm_model.summary())

print("Fit the model to check accuracy")
lstm_model.fit(X_train, y_train, batch_size=128, epochs=6, verbose=1, validation_split=0.2)

```



```

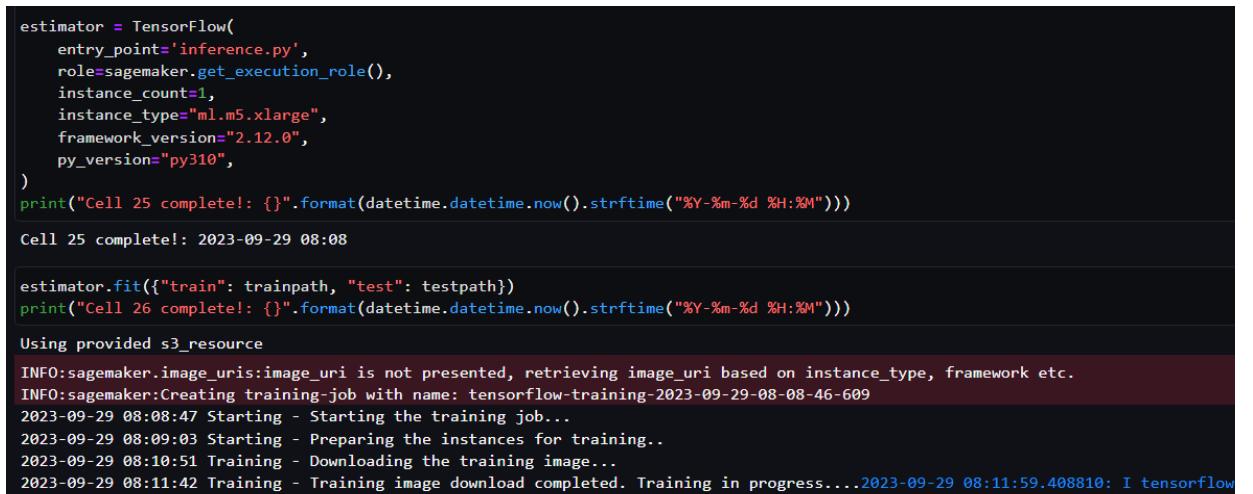
print("-----[TASK] Save Model in h5-----")
lstm_model.save(os.path.join(args.model_dir, "keras_model.h5"), save_format='h5')
print("Saved model in .h5 file")
print("-----\n")

print("-----[TASK] Save Tensorflow format-----")
version = 1
tf_model = tf.keras.models.load_model(os.path.join(args.model_dir, "keras_model.h5"))
tf_model.save(os.path.join(args.model_dir, str(version)))
print("Saved model in .pb file")
print("-----")

print("-----[TASK] Evaluate .h5 Accuracy-----")
score = lstm_model.evaluate(X_test, y_test, verbose=1)
print("Original Score:", score[0])
print("Original Accuracy:", score[1])
print("-----\n")

print("-----[TASK] Evaluate .pb Accuracy-----")
score = tf_model.evaluate(X_test, y_test, verbose=1)
print("Tensorflow Score:", score[0])
print("Tensorflow Accuracy:", score[1])
print("-----\n")

```



```

estimator = TensorFlow(
    entry_point='inference.py',
    role=sagemaker.get_execution_role(),
    instance_count=1,
    instance_type="ml.m5.xlarge",
    framework_version="2.12.0",
    py_version="py310",
)
print("Cell 25 complete!: {}".format(datetime.datetime.now().strftime("%Y-%m-%d %H:%M")))
Cell 25 complete!: 2023-09-29 08:08

estimator.fit({"train": trainpath, "test": testpath})
print("Cell 26 complete!: {}".format(datetime.datetime.now().strftime("%Y-%m-%d %H:%M")))

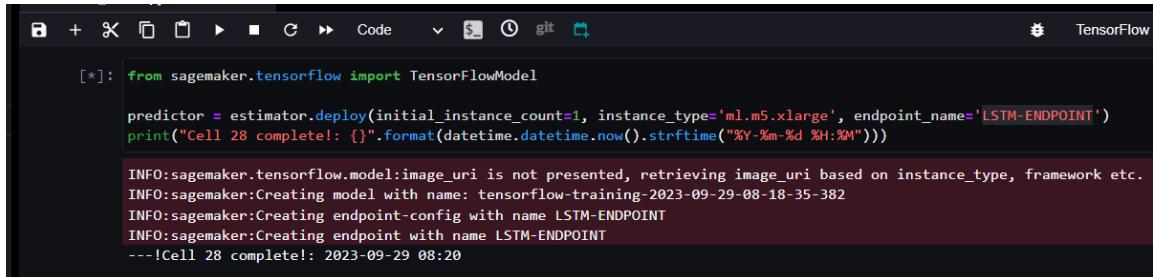
```

Using provided s3\_resource

```

INFO:sagemaker.image_uris:image_uri is not presented, retrieving image_uri based on instance_type, framework etc.
INFO:sagemaker:Creating training-job with name: tensorflow-training-2023-09-29-08-08-46-609
2023-09-29 08:08:47 Starting - Starting the training job...
2023-09-29 08:09:03 Starting - Preparing the instances for training..
2023-09-29 08:10:51 Training - Downloading the training image...
2023-09-29 08:11:42 Training - Training image download completed. Training in progress....2023-09-29 08:11:59.408810: I tensorflow

```



```

[*]: from sagemaker.tensorflow import TensorFlowModel

predictor = estimator.deploy(initial_instance_count=1, instance_type='ml.m5.xlarge', endpoint_name='LSTM-ENDPOINT')
print("Cell 28 complete!: {}".format(datetime.datetime.now().strftime("%Y-%m-%d %H:%M")))


```

```

INFO:sagemaker.tensorflow.model:image_uri is not presented, retrieving image_uri based on instance_type, framework etc.
INFO:sagemaker:Creating model with name: tensorflow-training-2023-09-29-08-18-35-382
INFO:sagemaker:Creating endpoint-config with name LSTM-ENDPOINT
INFO:sagemaker:Creating endpoint with name LSTM-ENDPOINT
---!Cell 28 complete!: 2023-09-29 08:20

```

```

endpoint_inputs = [
    "POSITIVE",
    "NEGATIVE",
    "NEUTRAL",
    "Love",
    "Happy!!!!!!",
    "Toy",
    "Disgrace",
    "GrACe",
    "Horse",
    "Programming",
    "Event",
    "Amazing",
    "Fantas tic",
    "L I K E",
    "LOV E",
    "Positive",
    "Negative",
    "Neutral"
]

for i in range(len(endpoint_inputs)):
    endpoint_inputs[i] = custom.preprocess_text(endpoint_inputs[i])

unseen_tokenized_endpoint = loaded_tokenizer.texts_to_sequences(endpoint_inputs)

unseen_padded_endpoint = pad_sequences(unseen_tokenized_endpoint, padding='post', maxlen=100)
print(unseen_padded_endpoint)

[[ 992     0     0 ...     0     0     0]
 [1390     0     0 ...     0     0     0]
 [9178     0     0 ...     0     0     0]
 ...

```

```

import boto3
import json
import numpy as np

# Initialize SageMaker runtime client
sagemaker_runtime = boto3.client('sagemaker-runtime')

# Define the endpoint name
endpoint_name = 'LSTM-ENDPOINT'

# Assuming unseen_padded_endpoint is a NumPy array
# Convert it to a list for JSON serialization
input_data = unseen_padded_endpoint.tolist()

# Call the SageMaker endpoint
response = sagemaker_runtime.invoke_endpoint(
    EndpointName=endpoint_name,
    ContentType='application/json',
    Body=json.dumps(input_data)
)

# Parse the response
result = json.loads(response['Body'].read().decode())

# Print or use the 'result' as needed
print(result)

{'predictions': [[0.841252923], [0.644867182], [0.743375838], [0.860342801], [0.856160879], [0.771649659], [0.434786558], [0.793271244], [0.8653], [0.777950823], [0.936216474], [0.786071897], [0.799581707], [0.765889287], [0.841252923], [0.644867182], [0.743375838]]}

```

Αφού θέσουμε κατάλληλες παραμέτρους για το πραγματοποιηθεί η εκπαίδευση του μοντέλου στο directory του Sagemaker (δημιουργείται αυτόματα στα S3 buckets), παράγεται ένα συμπιεσμένο αρχείο .tar.gz που μέσα έχει το μοντέλο.

Για τη χρήση του μοντέλου στο Web application αρκεί ένα Lambda Function που θα εκτελεί κώδικα όπως του παραπάνω screenshot για να παράγει τις αξιολογήσεις, οι οποίες με τη σειρά τους θα αποθηκεύονται σε έναν πίνακα float[].

Ωστόσο πρέπει να αναφέρω ότι το μοντέλο του AWS δεν έχει υλοποιηθεί ακόμα στο project γιατί υπάρχει ακόμα το πρόβλημα της προεπεξεργασίας. Το μοντέλο μέσω endpoint λειτουργεί, όμως μπορεί να δεχτεί μόνο padded sequences ως εισόδους, έτσι

λοιπόν πρέπει να βρεθεί τρόπος το μοντέλο να δέχεται strings ως εισόδους και να ξέρει από μόνο του πως να κάνει την προεπεξεργασία. Αν βρεθεί λύση σε αυτό το πρόβλημα το S.T.W. πλέον δεν θα έχει πια αδυναμίες και μπορεί σε πολύ λίγο χρόνο να αξιολογήσει τεράστια πλήθη σχολίων.

## Τι πετύχαμε

- Κατασκευή και αποθήκευση μοντέλου μηχανικής μάθησης LSTM
- Χρήση του μοντέλου σε Python και C#
- Χρήση της Μηχανικής Μάθησης σε web applications
- Εμβάθυνση στο κομμάτι της προεπεξεργασίας χάρης τον πειραματικό κώδικα C#
- Συνδυασμός C# και Python
- Ανακάλυψη του AWS
- Χρήση δεδομένων και μεθόδων AWS
- Κατασκευή και αποθήκευση μοντέλων μηχανικής μάθησης AWS
- Χρήση μοντέλων AWS

## Πηγές γνώσεων

Οι νέες γνώσεις που απέκτησα αφορούσαν κυρίως

- C#
- JavaScript
- OpenAI
- ASP.NET Framework
- Amazon Web Services
- Machine Learning
- Sentiment Analysis
- Python Scripts
- XML

Σε ό,τι απορία μπορεί να είχα, τα μέλη της ομάδας μου με στήριξαν και μου τις έλυσαν με πολύ σαφείς εξηγήσεις και με βάση δικές τους εμπειρίες.

Για την εξοικείωση των παραπάνω χρησιμοποιήσα πολλαπλές πηγές:

- Stack Overflow
- Amazon Web Services Guides
- OpenAI Guides
- Tensorflow

Στήριγμα αποτέλεσαν και τα παρακάτω μαθήματα από το πανεπιστήμιο τα οποία έχω παρακολουθήσει:

- Εισαγωγή στον Προγραμματισμό Υπολογιστών (Python)
- Εισαγωγή στον Προγραμματισμό με C++

- Εισαγωγή στην Επιστήμη Υπολογιστών
- Τεχνητή Νοημοσύνη
- Ανάπτυξη Εφαρμογών Πληροφοριακών Συστημάτων

## Προβλήματα

### Διακοπή ρεύματος

Μία εργάσιμη ημέρα κατά τη διάρκεια του πρώτου μήνα πρακτικής, λόγω έργων της ΔΕΗ, υπήρχε διακοπή ρεύματος για κάμποσες ώρες και δεν μπορούσαμε να χρησιμοποιήσουμε τους υπολογιστές μας. Τις ώρες αυτές τις εκμεταλλευτήκαμε για να συζητήσουμε διάφορα θέματα πάνω σε ό,τι καθήκοντα είχαμε. Εκείνη την ημέρα, κατέληξα στην ιδέα του Eventora S.T.W. την οποία και έγκρινε ο επιβλέπων μου.

### Σύνδεση στο διαδίκτυο

Κάποιες φορές, η σύνδεση στο διαδίκτυο μπορεί να έπεφτε για πολύ λίγο χρονικό διάστημα. Το συγκεκριμένο ζήτημα επιλύθηκε από την εταιρεία.

### Συσκευή

Τον τρίτο μήνα της πρακτικής μου, ο υπολογιστής πάνω στον οποίο δούλευα έπαθε υπερθέρμανση και δεν άνοιγε. Η δουλειά μου δεν χάθηκε αφού ήταν αποθηκευμένη στον δίσκο SSD και εκείνη την περίοδο έφτιαχνα το μοντέλο SageMaker μέσω της online πλατφόρμας, συνεπώς μπορούσα να συνεχίσω σε άλλον υπολογιστή. Ο υπολογιστής έφτιαξε εντός ολίγων ημερών με μία απλή αλλαγή του τζελ ψύξης στον επεξεργαστή.

### Επίλογος

Η πρακτική άσκηση ήταν μία ευχάριστη εμπειρία, γεμάτη με νέες προκλήσεις στις οποίες κατάφερα να ανταπεξέλθω τόσο με τις προηγούμενες γνώσεις μου όσο και με αυτές που απέκτησα σε αυτούς τους 3 μήνες.

Είμαι αρκετά ικανοποιημένος από τα δύο project που ανέλαβα καθώς με αυτά και έμαθα νέα πράγματα αλλά και δημιούργησα κάτι που θα μπορέσει να ενισχύσει την εταιρεία.

Θα ήθελα να εκφράσω και την ευγνωμοσύνη μου στην ΑΓΑΠΑΕΙ ΠΛΗΡΟΦΟΡΙΚΗ, τον επιβλέπων και την ομάδα μου που με δέχτηκαν ως δικό τους, με ενθάρρυναν να δώσω μορφή σε όσες ιδέες σκέψητηκα και με στήριξαν όσο περισσότερο μπορούσαν.

Πιστεύω πως η πρακτική άσκηση είναι μία ευκαιρία που πρέπει να αρπάξουν οι φοιτητές, όχι μόνο για να αποκτήσουν νέες γνώσεις στον τομέα τους, αλλά και να κάνουν τα πρώτα τους βήματα στον εργασιακό κόσμο.

## **Χρήσιμοι σύνδεσμοι**

Eventora: <https://www.eventora.com/>

Eventora help: <https://help.eventora.com/portal/el/kb/eventora>

Amazon Web Services: <https://aws.amazon.com/>

OpenAI: <https://openai.com/>

OpenAI Models: <https://openai.com/pricing>

OpenAI Tokenizer: <https://platform.openai.com/tokenizer>