Habr стаття: Автоматическое определение тональности текста (Sentiment Analysis)(<https://habr.com/ru/post/263171/>)

Головна ціль даної статі навчити користувача читан може зустрітисяча визначати тональність тексту, а також розказати про трудності з якими він може спіткнутися: “Цель этой задачи состоит в определении, является ли данный текст (допустим обзор фильма или комментарии) положительным, отрицательным или нейтральным по своему влиянию на репутацию конкретного объекта. ”, “Трудность анализа тональности заключается в присутствии эмоционально обогащенного языка — сленг, многозначность, неопределенность, сарказм, все эти факторы вводят в заблуждение не только людей, но и компьютеров. “

В першу чергу автор надає читачеві інформацію про те як буде працювати програма, а також визначає необхідні дані та говорить де їх можна знайти. Автор буде використовувати базу даних з сайту [www.kaggle.com](http://www.kaggle.com/), яка містить всю необхідну інформацію для нашої нейронної мережі.

Найшовши необхідну інформацію автор приступає до написання алгоритму.

Він розбив інформацію на кроки для кращого розуміння, тому далі я хотів би розібрати кожен його крок окремо.

### Шаг 1. Предобработка

На цьому кроці автор буде переробляти текст, для подальшого його використання.

В першу чергу він використовує бібліотеки Python для того щоб дістати чистий текст без лишньої інформації “Данная операция осуществляется с помощью библиотеки python — «Beautiful Soup».”.

Діставши текст автор позбавляється від іншої непотрібної інформації: “Также все числа и ссылки в тексте заменяются на тэги , . Далее в тексте присутствуют так называемые «стоп слова» — это частые слова в языке, которые в основном не несут никакую смысловую нагрузку (например, в английском языке это такие слова как «the, at, about…»). Стоп слова удаляются с помощью пакета Python Natural Language Toolkit (NLTK).”

В кінцеву результаті у нього получився масив слів які він буде використовувати в подальшому.

### Шаг 2. Представление в виде вектора

Тут автор хоче переробити дані в числа, для того щоб їх можна було використовувати в формулах. Він має два методи вирішення проблеми, проте я розгляну тільки один, адже він видає набагато кращий результат.

Метод полягає в використанні уже написаної нейромережі від Google Word2Vec, яка дозволяє знайти наскільки два слова подібні між собою. Далі ми розбиваємо уже відомі нам слова на кластери, і отримуємо двовимірний масив з інформацією про те якому кластеру належить кожне слово.

### Шаг 3. Классификация текстов

І останній крок це класифікація текстів. Автор не дає конкретної інформації про роботу, а використовує уже написані методи: ”Алгоритм классификаций Random Forest используется для классификаций документов в этом эксперименте. Алгоритм уже реализован в пакете scikit-learn, все, что нам остается это вскормить наши текстовые данные и указать количество деревьев. Дальше алгоритм все берет на себя, тренируется на обучающей выборке, сохраняет все необходимые данные.”

# Обучаем компьютер чувствам (sentiment analysis по-русски)(<https://habr.com/en/post/149605/>)

Незважаючи на похожість даного завдання з завданням з попередньої статті, автор використовує кардинально інший метод у вирішені даної проблеми.

“Целью *анализа тональности* является нахождение мнений в тексте и определение их свойств. В зависимости от поставленной задачи нас могут интересовать разные свойства, например:

1. автор — кому принадлежит это мнение
2. тема — о чем говорится во мнении
3. тональность — позиция автора относительно упомянутой темы (обычно «положительная» или «отрицательная») ”

Ми можемо зразу замітити, що в даній статі використовується більше інформації для визначення, наприклад тут ми дістаємо автора і тему статті, що допоможе нам визначити тональність набагато краще.

Спочатку автор розказує про методи, якими ми можемо зберігати дані, а також розказує про їх переваги та недоліки.

Після чого він будує власний класифікатор.

Для кожного слова він визначає наскільки позитивним воно є і просумувавши ми дістаємо інформацію про текст. Крім того автор показав роботу багатьох різних методів і показав їхню ефективність. За результатами дослідження він показав, що найкраще для

даної задачі використовувати TF-IDF метод.

Побудувавши його він розказує де ми можемо його використати:”В качестве примера практического применения классификатора тональности, я реализовал на скорую руку классификатор твитов, схема работа которой следующая:

1. производим поиск в Твиттере по названию фильмов
2. пропускаем твиты через классификатор тональности
3. получаем положительные и отрицательные высказывания о фильмах из Твиттера

“

# How to solve 90% of NLP problems: a step-by-step guide*(https://blog.insightdatascience.com/how-to-solve-90-of-nlp-problems-a-step-by-step-guide-fda605278e4e)*

Перш за все [Emmanuel Ameisen](https://blog.insightdatascience.com/@emmanuelameisen?source=post_page-----fda605278e4e----------------------) розказує про головні цілі даної статті, він хоче навчити читача трьом головним вмінням, які потрібні для виконання більшость завдань пов’язаних з NLP:

* Gather, prepare and inspect data
* Build simple models to start, and transition to deep learning if necessary
* Interpret and understand your models, to make sure you are actually capturing information and not noise

# Step 1: Gather your data

Кожна проблема машинного навчання починається з пошуку правильних датасетів саме тому автор в першу чергу говорить наякі класи поділяються датасети, та як читачеві розібрати які дані і коли йому потрібно використовувати.

Далі він переходить на конкретне завдання, а іменно визначення постів, які написані про реальну катастрофу.

Першим завданням, яке перед нам стало це правильно позначити датасет, адже інакше ми не зможемо правильно навчити нашу модель.

# Step 2: Clean your data

*The number one rule we follow is: “Your model will only ever be as good as your data.”*

Перш за все автор говорить про необхідність чистки данних, адже це дозволить не тільки швидше натренувати вашу модель, а також позбавить її від не потрібної інформації яка в майбутньому може не дати програмі працювати правильно.

# Step 3: Find a good data representation

Третім завданням, яке постала перед нами це визначення як ми будем подавати данні нашій моделі.

One-hot encoding (Bag of Words) – цей метод полягає в підрахунку кількості слів для кожного речення і записуання їх в словник, проте основною його проблемою є необхідність зберігання фнформації про кожне слово для кожного повідомлення, що зробить нам масив, який в більшості складається з 0.

# Step 4: Classification

Після того, як ми оброли наші дані ми можемо перейти до побудови моделі. Ми почнемо з побудови найпростішої Логістичної регресії, яка надасть нам точність в понад 75%, проте це досить мало.

# Step 5: Inspection

Тут нам необхідно проаналізувати роботу нашої моделі, а також визначити чому винукають конкретны помилки і як їх можна уникнути.

# Step 6: Accounting for vocabulary structure

## TF-IDF – цей метод дозволить нам позбутися зайвих слів, наприклад ті які, появляються надто часто, а також розтавить ваги для кожного слова.

Використовуючи цей метод, ми змогли дістати точність 76.2%, це невелике покращення проте воно допомогло нам краще знайти слова, які можуть передувати катастрофам, що зможе покращити наш результат при використанні більш складних моделей.

# **Step 7: Leveraging semantics**

# **Word2Vec –** ця модель допоможе нам працювати зі слова, які ми не зустрічали раніше, адже вона дозволяє зрозуміти наскільки два слова похожі між собою за значеннями.

Тепер ми змогли покращити результат до 77.7%. Уже непогано, проте можна і краще.

LIME – це відкрита бібліотека, яка дозволяє користувачу легше зрозуміти, як працює модель, у нашому випадку вона покаже, які слова ми викинули, а також наскільки кожне слово повпливало на результат.

# **Step 8: Leveraging syntax using end-to-end approaches**

Тут ми використовуємо [Convolutional Neural Networks for Sentence Classification](https://arxiv.org/abs/1408.5882) – CNN частіше використовується при роботі з зображенням проте тут він нам також знадобиться, він хорошиий тим, що він бере до уваги ще порядок слів, наприклад він побачить різницю між “Alex eats plants” і “Plants eat Alex.”

Тепер ми отримали результат близько 80%, що набагато більше початкових 75%.