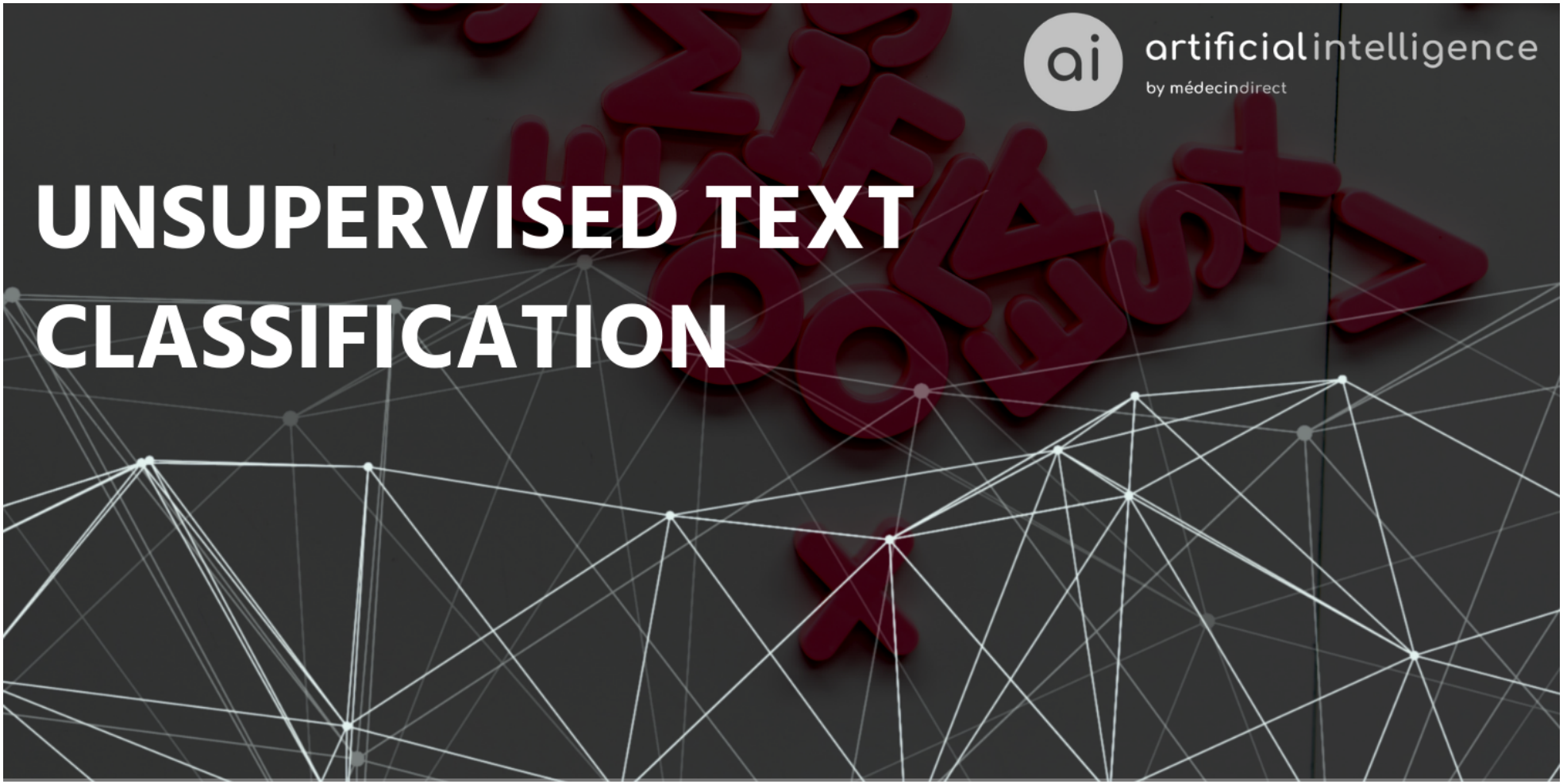

artificialintelligence
by médecindirect
UNSUPERVISED TEXT CLASSIFICATION

. . .

Unsupervised Text Classification

CONTEXT

Medium medecindirect.fr Follow
Jun 3, 2019 · 5 min read

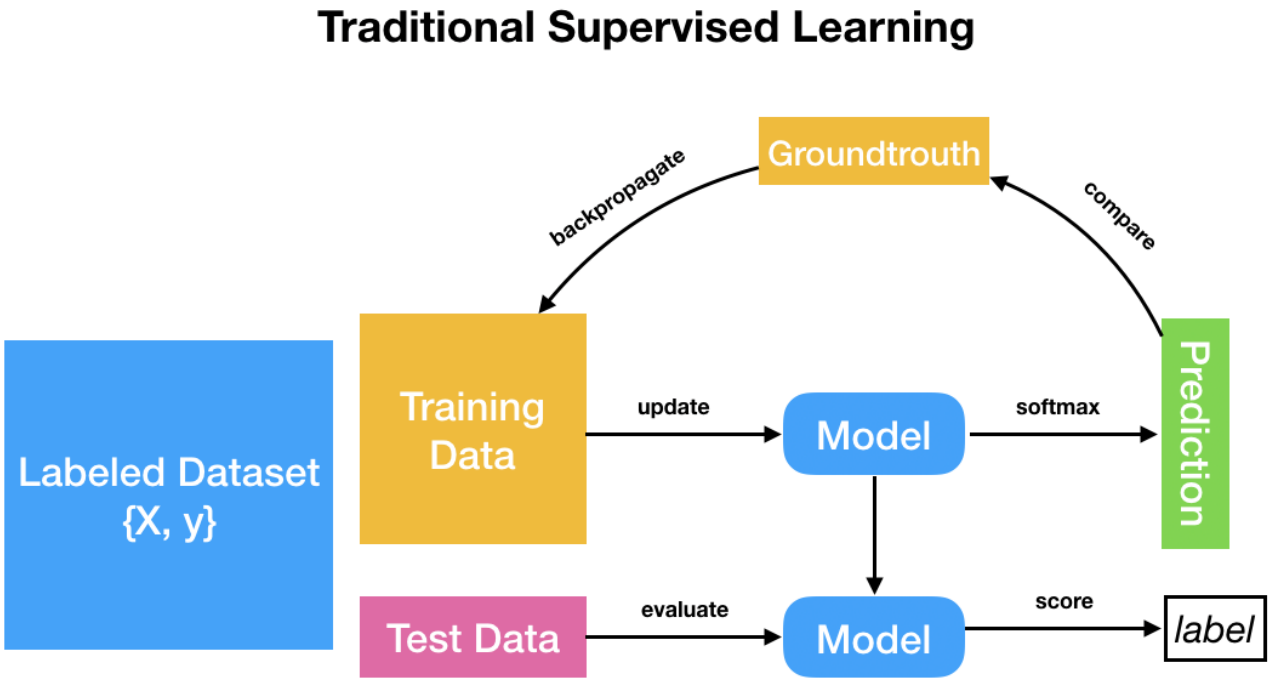
When I was a young boy and highly involved in the game of football, I asked my father when a player is offside? He gave me a short, yet simple description comparable to this definition:

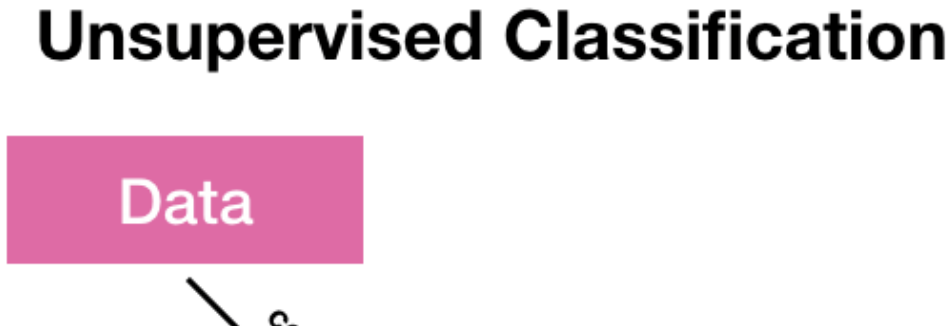
A player is in an offside position if: he is nearer to his opponents' goal line than both the ball and the second last opponent.

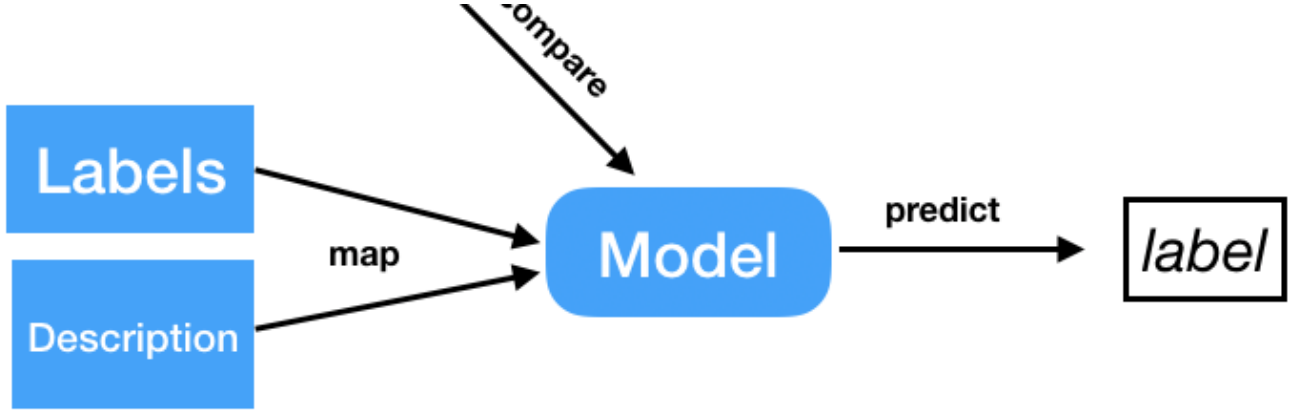
So instead of giving me thousands of examples or images of situations where a player is either in an on- or offside position, I understood the meaning of "offside" with a simple definition (and maybe some few, additional examples) and was able to distinguish between on- and offside in football in the future.

This phenomenon inspired me to apply the same concept to a text classification problem here at MédecinDirect. Instead of learning the mapping between text to label with many iterations over big training sets,

the model maps the description of a label to its label and is able to predict the label of incoming text without any training.

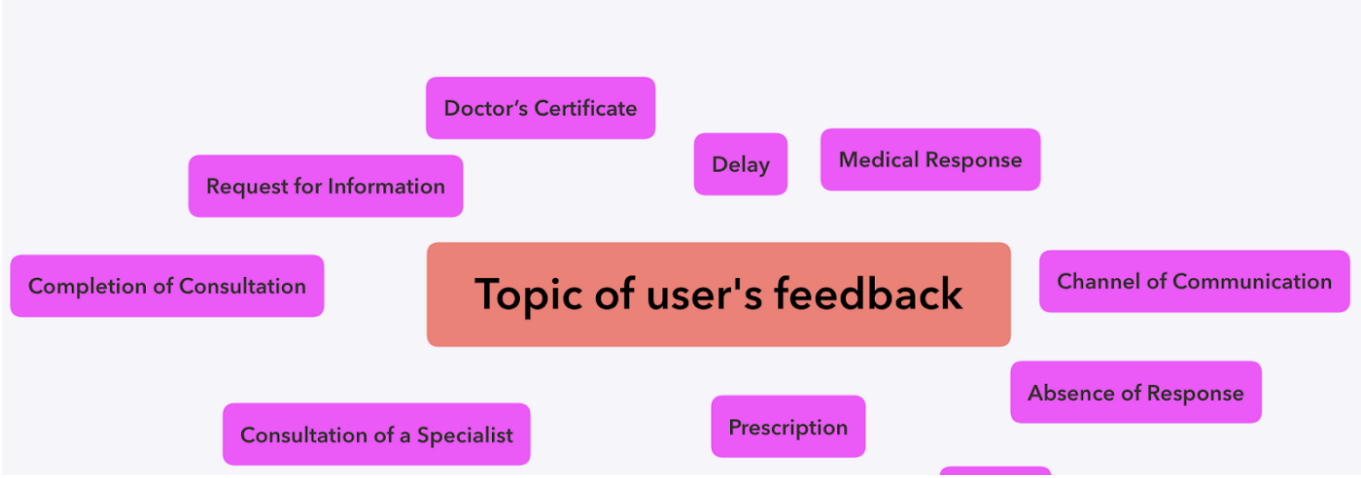


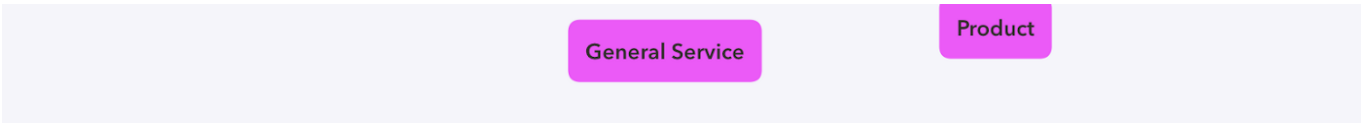




OBJECTIVE

In order to help the Customer Service team at MédecinDirect and speed up their response time to potential unsatisfied customers, we label user's feedback automatically into following 11 categories (these are examples and not the actual categories) :





PROBLEMS

My first intention was to solve this problem conventionally as a supervised classification problem and feed a model with training samples. However, I found that only roughly 200 of more than 5000 ratings were labeled. What made things even worse: the dataset is highly-skewed, with the least predominant class containing only 3 samples.

Since, we have 11 labels for prediction and a rating can have multiple labels at the same time, this is clearly waaaay too less data to make a precise and sophisticated prediction.

It is for those reasons, why I had to be a bit more creative concerning this issue.

MATHS BEHIND

The main reason why I was able to understand the offside problematic as a child quickly was most likely my general understanding and interest in football. My sister, which was more into other sports back then, didn't get a grasp on the offside problematic until today, I'm afraid.

In Natural Language Processing the general understanding of text is trained on massive corpus, such as all Wikipedia entries of a given language. As a result so called WordEmbeddings represent each word as a high dimensional vector.

As seen below, each token of the sentence: “Words represented as vectors.” is represented as a vector with 3072 dimensions. Additionally, this concept can be easily expanded to sentences (StackedEmbeddings) or entire documents (DocumentEmbeddings).

```
: # create a sentence and embed it
sentence = Sentence('Words represented as vectors.',use_tokenizer=True)
embedding.embed(sentence)
for token in sentence:
    print(token)
    print("{} - Vector with {} dimensions".format(token.embedding, len(token.embedding)))
    print("")

sentence_embedding.embed(sentence)
print("And one single vector for the sentence: {}".format(sentence.embedding))
```

```
Token: 1 Words
tensor([-0.2731, -0.0716,  0.5012,  ...,  0.8949,  0.9229,  0.1269]) - Vector with 3072 dimensions

Token: 2 represented
tensor([-0.5671, -0.2177,  0.3155,  ...,  0.7454,  0.6338,  0.0449]) - Vector with 3072 dimensions

Token: 3 as
tensor([-0.3648,  0.0270,  0.1370,  ...,  0.1018,  0.5720,  0.0651]) - Vector with 3072 dimensions

Token: 4 vectors
tensor([-0.5539, -0.4898, -0.0795,  ...,  0.5613,  0.8817,  0.3269]) - Vector with 3072 dimensions

Token: 5 .
tensor([-0.1765, -0.0554,  0.1779,  ..., -0.4395,  0.3309,  0.1104]) - Vector with 3072 dimensions

And one single vector for the sentence: tensor([-0.2419,  0.2743,  0.2511,  ...,  0.3728,  0.6683,  0.1348])
```

Similar words yield to similar vectors within these Embeddings.

A way to rate the similarity of two vectors is the so called cosine-distance:

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

If vector A and B are exactly similar, the cosine distance is 1. Viceversa it is -1, if they are totally opposite:

$$cos(\theta) = \frac{[2,2,2] \,.\, [2,2,2]}{\sqrt{12} * \sqrt{12}} = \frac{12}{12} = 1$$

$$cos(\theta) = \frac{[-2,-2,-2] \,.\, [2,2,2]}{\sqrt{12} * \sqrt{12}} = \frac{-12}{12} = -1$$

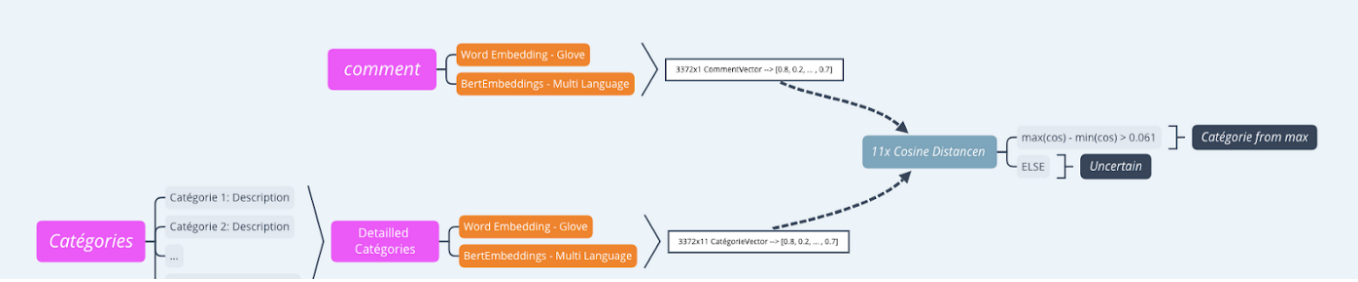
APPROACH

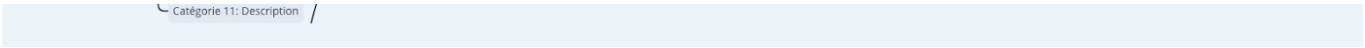
The basic idea is to create a label vector, which represents the 11 labels as vectors and calculate the cosine distance of each label (A) to the user's feedback (B).

The label with the highest similarity or is higher than a certain threshold will be assigned to the user rating.

The single words of the label are quite unspecific in this particular case, since the label "product" for example corresponds to many different things in the real world. Thus a label description, similar to the offside definition in the introduction, is used to represent the label numerically as a vector.

Since there are some architectural choices to pick, such as the DocumentEmbedding type, which kind of description to use for the labels, how confident the model needs to be, etc.. the labeled 200 samples are used to evaluate the best performing architecture for this problem.





The architecture in use takes for the label description three synthetic examples per label and constructs with a stacked DocumentEmbedding a *CategorieVector* for each label. The same DocumentEmbedding is used to transform the user's feedback into one *CommentVector*. In total 11 different cosine distances are computed. If the model has a clear favourite among them it outputs the label with the maximum value (yes, currently no multi-label classification), else it outputs the 12th label: "Uncertain".

CONCLUSION & OUTLOOK

Instead of learning a mapping function between text to label and construct a supervised classification model in order to make a prediction, this article proposes a shift of attention to the "meaning" behind the label. This has two main benefits in respect to traditional approaches:

Firstly, the model learns in a more human-like way, by understanding the actual meaning of each category it shall predict. Secondly, this approach can be implemented with no available data at hand, since the WordEmbeddings are publically available and pretrained. This allows immediate results and is super beneficial, when there is a lack of (good) data.

The drawback in this particular example is of course, that there is no testing data at all to evaluate the actual performance of the model before usage. Therefore I set the confidence level quite high in order to avoid incorrectly classifications of the user's feedback. This ensures that the model only predicts a class if it is strongly "convinced".

In the upcoming months, we will combine this approach with reinforcement learning techniques to improve the model's prediction accuracy over time. We'll keep you posted!

. . .

ABOUT THE AUTHOR

Tim-Noah SCHMIDT : Data Science Intern — Médecin Direct
tim-noah.schmidt@medecindirect.fr —https://github.com/timnoah

Machine Learning Healthcare Telemedicine

Stay up to date on coronavirus (Covid-19)

Follow the Medium Coronavirus Blog or sign up for the newsletter to read expert-backed coronavirus stories from Medium and across the web, such as:

- More people are getting infected with Covid-19, but fewer ventilators are needed. Why?
- Will people get the coronavirus vaccine?
- Hey America, what happened to contact tracing?