Text augmentation for Machine Learning tasks: How to grow your text dataset for classification?

Maâli Mnasri Follow
Jan 18, 2019 · 6 min read

Text classification is the task of tagging automatically text segments with predefined categories. It is a supervised process that learns from observing pre-labelled data. There are multiple known algorithms to achieve this goal like the well known Naive Bayes, SVM and the different Deep Neural Networks.

Text classification can be used for **spam detection, language recognition, sentiment analysis**, and the list is so long. @Opla.ai, we deal with text classification, for example, to perform **intent recognition** within our chatbot builder. Actually, having multiple intent categories, **intent classification** allows to assign the right intent to the user utterance.

Globally, text classification is no more a mysterious task especially using the most basic algorithms. You can easily find excellent examples on internet on how to do it. However, most of these articles/tutorials explain how to learn classification **HAVING** the training data. Usually, we can find standard training data for classification which is great to start with Text Classification and to compare models. Though, many of us are interested in this task because they have a particular need in their company/project. And that's when we start wondering about the data problem as **text classifiers are worthless without enough training data**. The quality of these also matters.

Sometimes, we have no data at all. So we need to create it or collect it. But other times, we do have data but not enough to achieve high accuracy classification. This problem arises in particular when dealing with languages other than English or also for uncommon classification labels. In this article we will try to answer partially the question of **how can we train a classification model on little data**? Answering this question is associated to the **Small AI** area of research which tries to apply machine learning models on small datasets. You can find more details on Small AI in our **article on future directions of AI**.

Data augmentation, quésaco?

It is the process of extending a set of existant data by performing on them some operations which depend on the data type. The goal is to create new labeled data from the existing ones. For example, when we deal with images we can create new data by performing operations like **rotation, translation, scaling up or down**, etc. Data augmentation can also be achieved by enriching the data using **external knowledge sources**.

When it comes to **text**, there are other ways to create new data and the possibilities are not few.

Approaches

Word/sentence shuffling

A simple baseline for data augmentation is **shuffling the text elements to create new text.** For example, if we have labeled sentences and we want to get more, we can shuffle each sentence words to create a new sentence. This option is only valid for classification algorithms that don't take into account the words order within a sentence. So in practice, we should tokenize each sentence into words. Then we shuffle those words and rejoin them to create new sentences.

```python
from nltk import word_tokenize
import random


def augment(sentence,n):
    new_sentences = []
    words = word_tokenize(sentence)
    for i in range(n):
        random.shuffle(words)
        new_sentences.append(' '.join(words))
    new_sentences = list(set(new_sentences))
    return new_sentences

if __name__ == '__main__':
    nsentences = augment("Data augmentation is a challenging process",10)
    for s in nsentences:
        print s
```

words_shuffling.py hosted with ♡ by GitHub view raw

Word replacement

This technique aims at replacing some words in the sentence with new words while preserving its meaning. For example, having the sentence "Do not park your car in this placement" we can create these new sentences by substituting the word car with another word each time:

"Do not park your vehicle in this placement"

"Do not park your motocar in this placement"

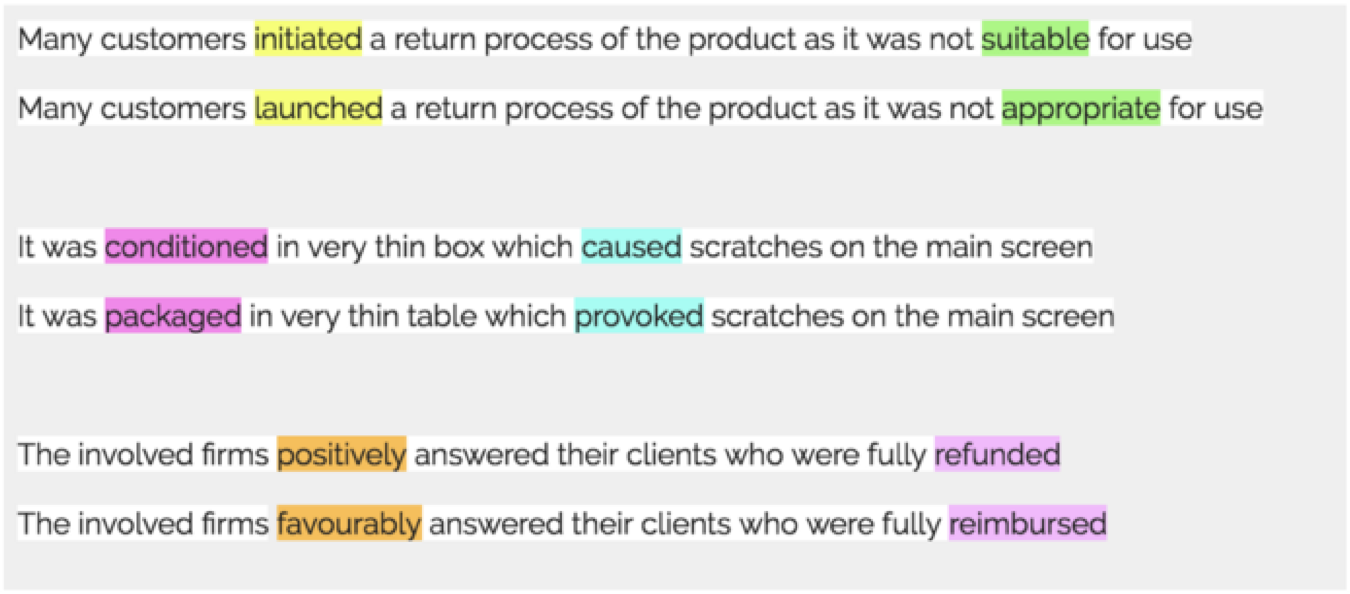
To replace words, there are several ways to find substituents. The most relevant way is to replace words by their **synonyms** and expressions by their **paraphrases**. The code snippet below allows to perform this. I used a paraphrase list from PPDB. You can find **all the code along with the used data here.**

```
from nltk import word_tokenize
from nltk.corpus import stopwords

stoplist = stopwords.words('english')


def get_synonyms_lexicon(path):
    synonyms_lexicon = {}
    text_entries = [l.strip() for l in open(path).readlines()]
    for e in text_entries:
        e = e.split(' ')
        k = e[0]
        v = e[1:len(e)]
        synonyms_lexicon[k] = v
    return synonyms_lexicon


def synonym_replacement(sentence, synonyms_lexicon):
    keys = synonyms_lexicon.keys()
    words = word_tokenize(sentence)
    n_sentence = sentence
    for w in words:
        if w not in stoplist:
            if w in keys:
                n_sentence = n_sentence.replace(w, synonyms_lexicon[w][0])  # we replace with th
    return n_sentence


if __name__ == '__main__':
    text = 'Many customers initiated a return process of the product as it was not suitable for
           'It was conditioned in very thin box which caused scratches on the main screen.' \
           'The involved firms positively answered their clients who were fully refunded.'
    sentences = text.split('.')
    sentences.remove('')
    print sentences
    synonyms_lexicon = get_synonyms_lexicon('./ppdb-xl.txt')
    for sentence in sentences:
        new_sentence = synonym_replacement(sentence, synonyms_lexicon)
        print '%s' % sentence
        print '%s' % new_sentence
        print '\n'
```

```
                print "\n"
```

word_replacement.py hosted with ♡ by GitHub view raw

We ran the code above and these are the results. The words highlighted with the same colors represent the original words and their synonym substitute.



Many customers initiated a return process of the product as it was not suitable for use

Many customers launched a return process of the product as it was not appropriate for use

It was conditioned in very thin box which caused scratches on the main screen

It was packaged in very thin table which provoked scratches on the main screen

The involved firms positively answered their clients who were fully refunded

The involved firms favourably answered their clients who were fully reimbursed

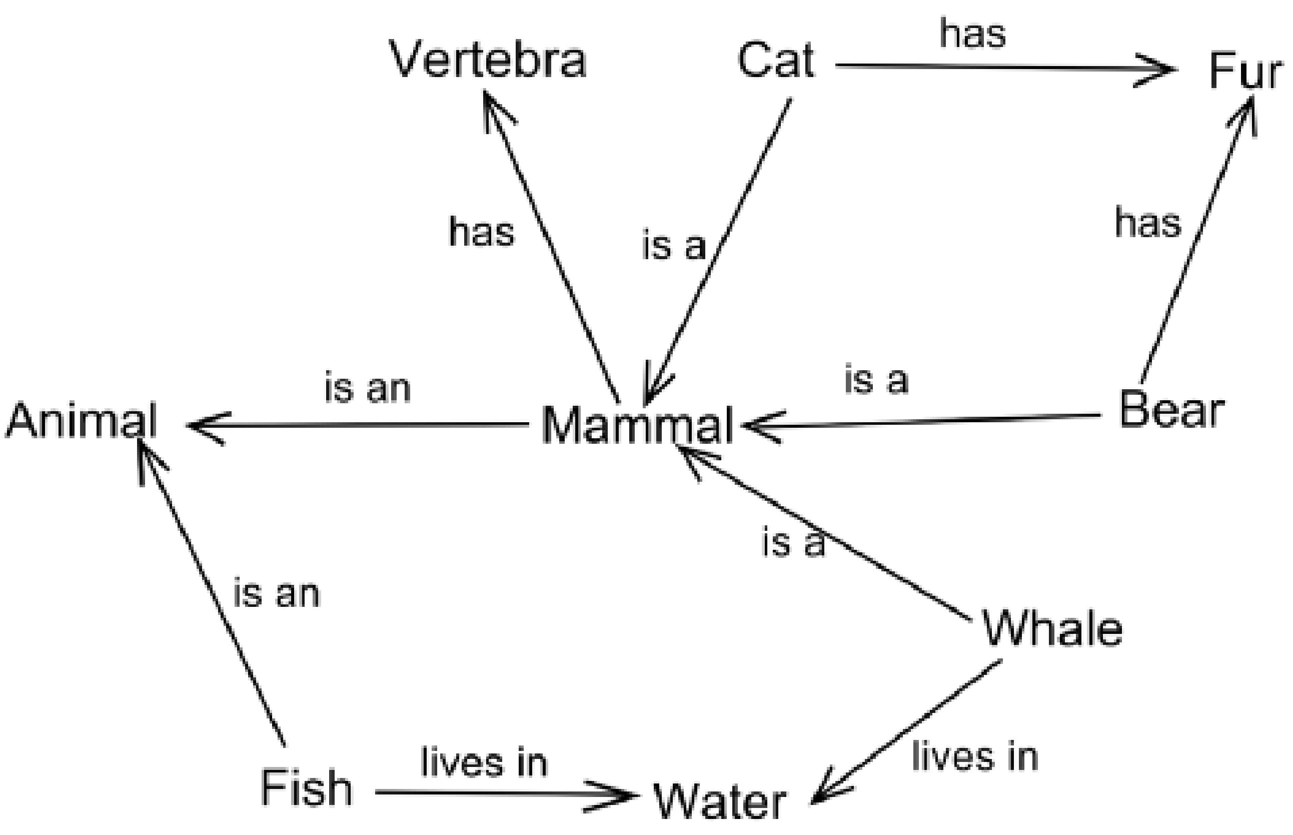


Figure 1: Example of a semantic network. Source :https://commons.wikimedia.org/w/index.php?curid=1353062

Semantic networks can also be used to find substituents. Concepts related by an “is a” relation in these knowledge bases can replace one another. For example, if we have this subgraph of a Semantic Net, we can replace the word “mammal” by the word “animal” as a mammal is an animal.

More sophisticated models have been proposed to augment text data by word replacement such as **predictive models** trained to predict the next word having the current one(Kobayashi, 2018).

Syntax-tree manipulation

This option allows to generate syntactic paraphrases of the source sentence. Many syntactic transformations can be applied to create paraphrases such as **moving from active to passive voice, breaking apart a sentence containing a subordinate clause**, etc. Three examples of syntactic paraphrasing are presented below. They are extracted from (Dras, 1997) article on using synchronous TAGs for paraphrasing.

(1) a. The salesman made an attempt to wear Steven down.

(1) b. The salesman attempted to wear Steven down.

(2) a. The compere who put the contestant to the lie detector gained the cheers of the audience.

(2) b. The compere put the contestant to the lie detector test. He gained the cheers of the audience.

(3) a. The smile broke his composure.

(3) b. His composure was broken by the smile.

This operation requires, first, to generate for each sentence its syntactic dependency tree. Then, use some rules to make transformations on that tree while preserving the meaning and the grammatical quality. The graph below shows an example of the source sentence syntactic tree as well as the new sentences ones.

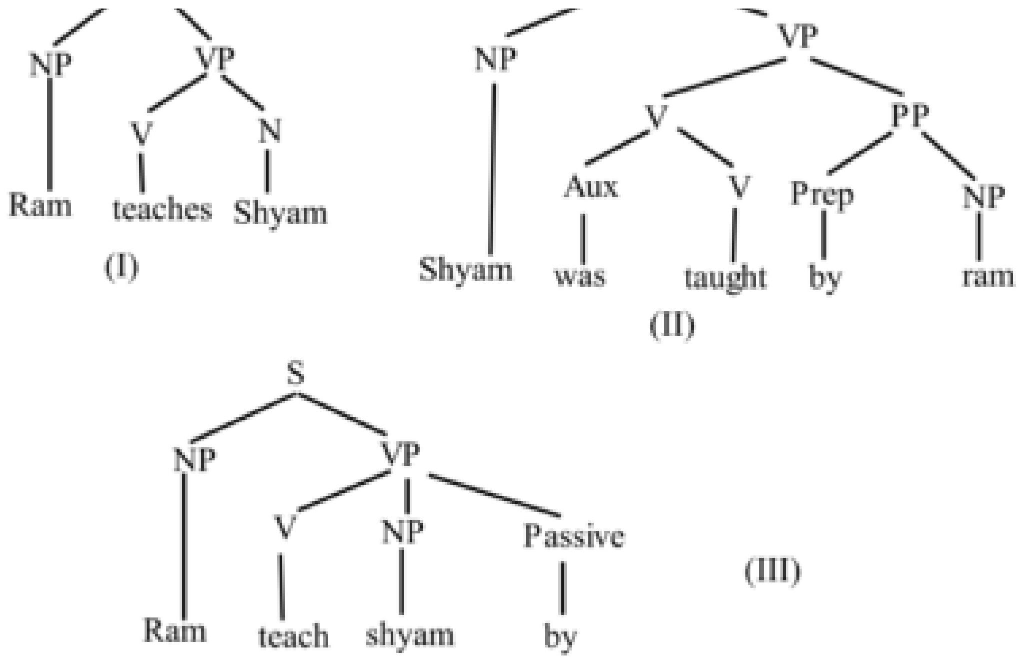


Figure 2 : Transformational Grammar (Chowdhari, 2012).

While the paraphrases are often of good quality, **this NLP task is computationally expensive** which can be inconvenient in practice.

Query expansion

Another approach to text data augmentation is the so called query expansion technique. This method aims at finding new sentences similar to a given sentence **by considering this latter as query and searching for its answer.** Using a generic or a domain specific search engine, we can define a query as our original sentence. Then, we select **the most relevant search results as new sentences**. These new sentences should inherit the same tag as the original sentence (the query). Sentence query expansion can also be performed by requesting knowledge bases such as Wikipedia or WordNet rather than search engines.

Conclusion

In machine learning systems, the more data your system observes the more accurate it gets. So, whatever the technique you choose, data augmentation should increase your classifier accuracy if it is done correctly. From another perspective, data augmentation methods do not only enlarge your training set, but also lead to **more generalizable classifiers.** Data augmentation has also been **identified among the best practices** when working with convolutional neural networks applied to document analysis (Simard & al., 2003).

Despite the ease of some of the presented approaches, it is undeniable that language understanding is categorically harder than image understanding from an algorithmic perspective. This is not necessarily a bad news as it means **there's more green field to explore in this area**. That's why we keep this subject among our high priority points of interest in Opla's R&D team.

Opla/SmallData-Augmentation-MachineLearning

Text Augmentation for Machine Learning tasks. Small data: How to grow your text dataset for classification ? ...

github.com

CHOWDHARY, K. R. Natural Language Processing. MBM Engineering College, Jodhpur, India, lecture notes, 2012.

DRAS, Mark. Representing paraphrases using synchronous tags. In : Proceedings of the eighth conference on European chapter of the Association for Computational Linguistics. Association for Computational Linguistics, 1997. p. 516–518.

KOBAYASHI, Sosuke. Contextual Augmentation: Data Augmentation by Words with Paradigmatic Relations. arXiv preprint arXiv:1805.06201, 2018.

SIMARD, Patrice Y., STEINKRAUS, Dave, et PLATT, John C. Best practices for convolutional neural networks applied to visual document analysis. In : null. IEEE, 2003. p. 958.

Machine Learning NLP Classification Training Data Small Data