How to improve the performance of a neural network with unbalanced data for text classification in insurance application

Isaac Cohen Sabban, Olivier Lopez, Yann Mercuzot

Virtual Colloquium 2020

May 2020

Summary

Goal : Prediction of the evolution of a claim

- Use artificial intelligence to early identify claims that require more attention

- Explore and find a model to deal with the unbalanced characteristic

Summary

1. Goal
2. Neural Networks
   - Pre processing : N-Grams and The Embedding matrix
   - Convolutional Neural Network : CNN
   - LSTM
3. Rebalancing of the dataset
4. Results

N-Grams

Definition

An n-gram is a contiguous sequence of n items from a given sample of text or speech.

Example

"client hits a pedestrian on a protected passage, shock on the fender, to the bonnet, the pedestrian is injured" .

1-Grams"client" "hits" "a" "pedestrian" "on" "a" "protected" "passage" "shock" "on" "the" "fender" "to" "the" "bonnet"

2-Grams"client hits" "hits a" "a pedestrian" "the pedestrian" "pedestrian is" "is injured"

N-Grams helps us to catch the context

How does it works ?

Each claim is composed by sentences to describe the claim circumstances, two representations are possible :

1. Associate a unique numerical value in order to transform our textual information into numerical values, exactly as Key-Value system creates a vector of values.
2. Transform the sentence into a matrix encode by the One Hot transformation

Example of the content of a claim

"client hits a pedestrian on a protected passage, shock on the fender, to the bonnet, the pedestrian is injured"

This sentence after the pre-processing step become :

Values vector

[1 22 5 2 . . .]

One hot Encoding Matrix

$$\underbrace{\begin{pmatrix} 1 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 1 & \cdots & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 \end{pmatrix}}_{\text{sentence size}} \left.\vphantom{\begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}}\right\} \text{dictionary size}$$

Limitations

These representations are limited because :

1. The Dictionary could be very large
2. Every pair of entities has the same distance.

A better representation exists : The Embedding Matrix

Embedding Matrix

Definition

An embedding matrix is a linear mapping from the original space (one-of-k) to a real-valued space where entities can have meaningful relationships.

Advantages :

- Dimensional Reduction
- Takes into account the context

The perfect input for a Neural Network

Convolutional Neural Network : CNN

CNN performs processing sequence, each step is usually called a layer. Different kind of layer exist:

- Convolution layer
- Pooling layer
- Normalization layer
- Fully Connected layer
- Loss layer

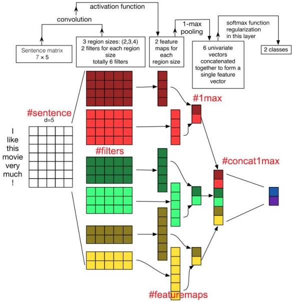


Figure: Kim CNN

Convolution Layer

A : Input

B : Filter or Feature Detector

C : Feature map

$$\underbrace{\begin{bmatrix}1&1&1&1&1&1&1&1\\0&1&1&1&1&1&1&1\\0&0&1&1&1&1&1&1\\0&0&0&1&1&1&1&1\\0&0&0&0&1&1&1&1\\0&0&0&0&0&1&1&1\\0&0&0&0&0&0&1&1\\0&0&0&0&0&0&0&1\end{bmatrix}}_{n} * \underbrace{\begin{bmatrix}1&0&1\\0&1&0\\1&0&1\end{bmatrix}}_{f} = \begin{bmatrix}\ &\ &\ &\ &\ &\ \\ \ &\ &\ &\ &\ &\ \\ \ &\ &\ &\ &\ &\ \\ \ &\ &\ &\ &\ &\ \\ \ &\ &\ &\ &\ &\ \\ \ &\ &\ &\ &\ &\ \end{bmatrix}$$

$\forall\, i,j$

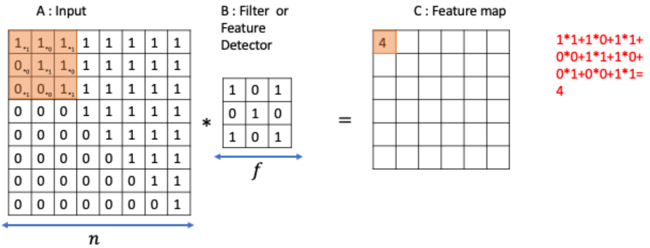
$$c_{ij} = \sum_{u=0}^{fr} \sum_{v=0}^{fc} a_{i+uj+v} b_{uv}$$

$fr =$ *row number of* B

$fc =$ *column number of* B

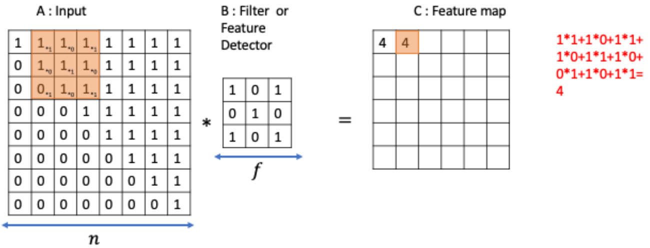
Size of C is given by $\left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor \times \left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor$ $\begin{cases} f = \textit{filter size} \\ p = \textit{padding} \\ s = \textit{stride} \end{cases}$

Convolution Layer



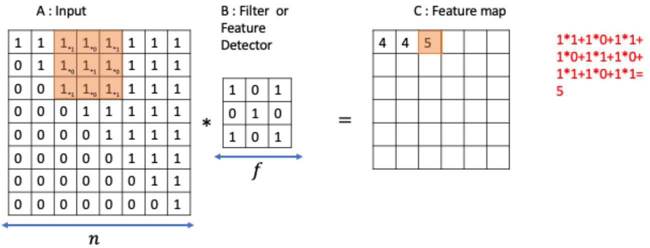
Size of C is given by $\left\lfloor \frac{n+2p-f}{s}+1 \right\rfloor \times \left\lfloor \frac{n+2p-f}{s}+1 \right\rfloor \quad \begin{cases} f = filter\ size \\ p = padding \\ s = stride \end{cases}$

Convolution Layer



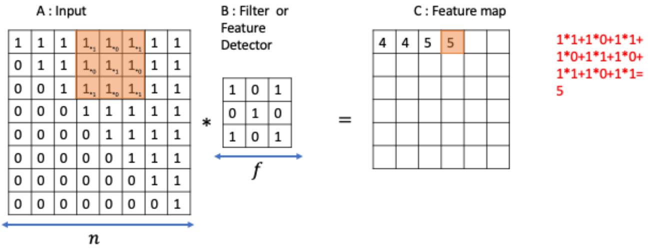
Size of C is given by $\left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor \times \left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor$ $\begin{cases} f = \textit{filter size} \\ p = \textit{padding} \\ s = \textit{stride} \end{cases}$

Convolution Layer



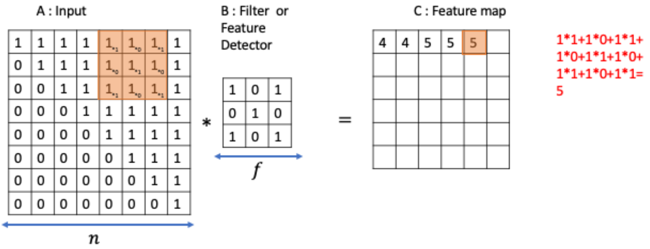
Size of C is given by $\left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor \times \left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor$ $\begin{cases} f = \textit{filter size} \\ p = \textit{padding} \\ s = \textit{stride} \end{cases}$

Convolution Layer



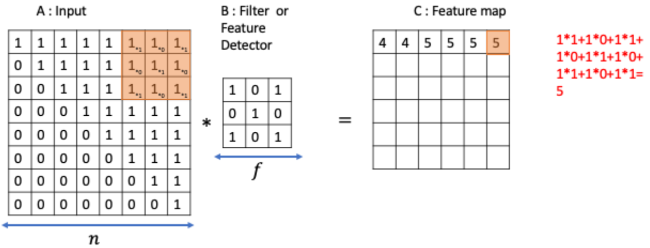
Size of C is given by $$\left\lfloor \frac{n+2p-f}{s}+1 \right\rfloor \times \left\lfloor \frac{n+2p-f}{s}+1 \right\rfloor \quad \begin{cases} f = filter\ size \\ p = padding \\ s = stride \end{cases}$$

Convolution Layer



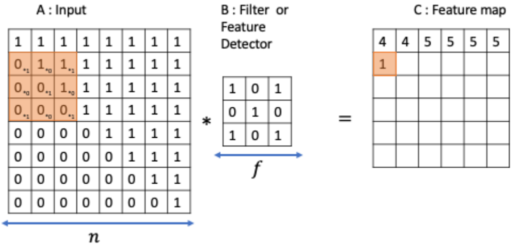
Size of C is given by $\left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor \times \left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor$ $\begin{cases} f = filter\ size \\ p = padding \\ s = stride \end{cases}$

Convolution Layer



Size of C is given by $\left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor \times \left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor$ $\begin{cases} f = \text{filter size} \\ p = \text{padding} \\ s = \text{stride} \end{cases}$

Convolution Layer



0*1+1*0+1*1+
0*0+0*1+1*0+
0*1+0*0+0*1=
1

Size of C is given by $\left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor \times \left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor$ $\begin{cases} f = filter\ size \\ p = padding \\ s = stride \end{cases}$

Convolution Layer

A : Input

B : Filter or Feature Detector

C : Feature map

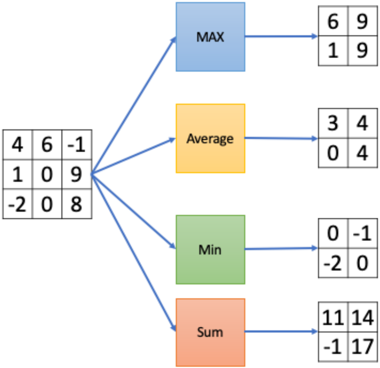
$$
\underbrace{\begin{array}{|c|c|c|c|c|c|c|c|}
\hline
1&1&1&1&1&1&1&1\\ \hline
0&1&1&1&1&1&1&1\\ \hline
0&0&1&1&1&1&1&1\\ \hline
0&0&0&1&1&1&1&1\\ \hline
0&0&0&0&1&1&1&1\\ \hline
0&0&0&0&0&1_{\times1}&1_{\times0}&1_{\times1}\\ \hline
0&0&0&0&0&0_{\times0}&1_{\times1}&1_{\times0}\\ \hline
0&0&0&0&0&0_{\times1}&0_{\times0}&1_{\times1}\\ \hline
\end{array}}_{n}
*
\underbrace{\begin{array}{|c|c|c|}
\hline
1&0&1\\ \hline
0&1&0\\ \hline
1&0&1\\ \hline
\end{array}}_{f}
=
\begin{array}{|c|c|c|c|c|c|}
\hline
4&4&5&5&5&5\\ \hline
1&4&4&5&5&5\\ \hline
1&1&4&4&5&5\\ \hline
0&1&1&4&4&5\\ \hline
0&0&1&1&4&4\\ \hline
0&0&0&1&1&4\\ \hline
\end{array}
$$

1*1+1*0+1*1+
0*0+1*1+1*0+
0*1+0*0+1*1=
4

Size of C is given by

$$\left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor \times \left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor \quad \begin{cases} f = filter\ size \\ p = padding \\ s = stride \end{cases}$$

Pooling Layer



$$\left\lfloor \frac{n-f}{s} + 1 \right\rfloor \times \left\lfloor \frac{n-f}{s} + 1 \right\rfloor$$

Figure: Pooling Step

CNN and Text

Client	0,1	0,9	0,5	0	0	0
hits	0,7	0,1	0	0	0	0
a	0	0	0	0	0	0
...	.	.	.	.	.	.
is	0	0	0	0	0	0
injured	0,8	0,1	0	0	0	0

Embedding dimension

Transpose of the Embedding Vector associate to the word « client »

CNN and Text

Client	0,1	0,9	0,5	0	0	0	Word
hits	0,7	0,1	0	0	0	0	
a	0	0	0	0	0	0	
...	.	.	.	.	.	.	
is	0	0	0	0	0	0	
injured	0,8	0,1	0	0	0	0	

CNN and Text

Client	0,1	0,9	0,5	0	0	0
hits	0,7	0,1	0	0	0	0
a	0	0	0	0	0	0
...	.	.	.	.	.	.
is	0	0	0	0	0	0
injured	0,8	0,1	0	0	0	0

Word

Client	0,1	0,9	0,5	0	0	0
hits	0,7	0,1	0	0	0	0
a	0	0	0	0	0	0
...	.	.	.	.	.	.
is	0	0	0	0	0	0
injured	0,8	0,1	0	0	0	0

2 grams

CNN and Text

Word

Client	0,1	0,9	0,5	0	0	0
hits	0,7	0,1	0	0	0	0
a	0	0	0	0	0	0
...	.	.	.	.	.	.
is	0	0	0	0	0	0
injured	0,8	0,1	0	0	0	0

2 grams

Client	0,1	0,9	0,5	0	0	0
hits	0,7	0,1	0	0	0	0
a	0	0	0	0	0	0
...	.	.	.	.	.	.
is	0	0	0	0	0	0
injured	0,8	0,1	0	0	0	0

3 grams

Client	0,1	0,9	0,5	0	0	0
hits	0,7	0,1	0	0	0	0
a	0	0	0	0	0	0
...	.	.	.	.	.	.
is	0	0	0	0	0	0
injured	0,8	0,1	0	0	0	0

Long Short-Term Memory

The Recurrent Neural Networks' main idea is that data are dependent on each other.

- RNNs consider an information sequence unlike CNNs
- Recurrent because they perform the same task for each element of a sequence.
- RNNs have a memory cell
- LSTMs are designed to avoid the long-term dependency problem.

Summary

Censorship and Kaplan Meier

We have a right censorship in our dataset because some claims are still going on.

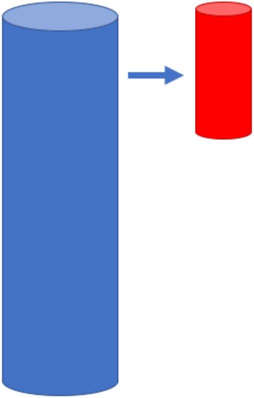
We use Kaplan Meier to correct censorship's bias.

Bagging

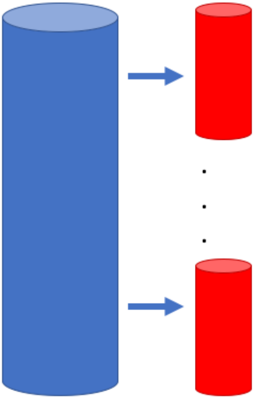
Definition

Bagging for bootstrap aggregation is a technique for reducing the variance of an estimated prediction function. It's seems to work especially well for high-variance, low-bias procedures, such as trees.

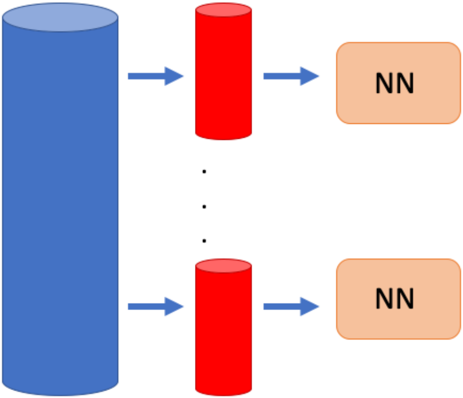
Bagging



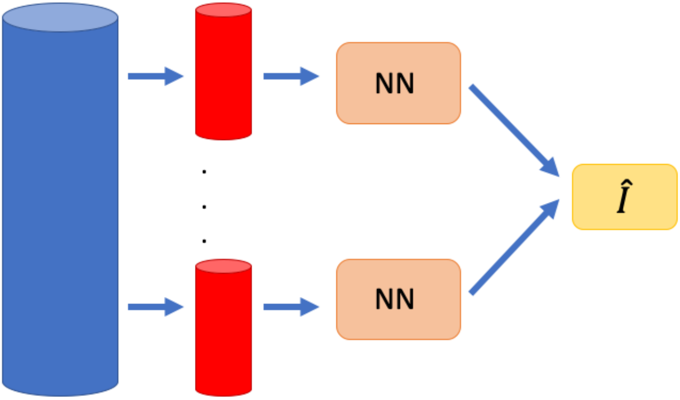
Bagging



Bagging



Bagging



Problems

In some cases we know how to generate data :

- structured data : SMOTE (Synthetic Minority Over-Sampling TEchnique)

- images : mirroring, random cropping, rotation, shearing, local warping, color shifting, distortions, etc

But these techniques are not usable for text data

Balanced

Let :

- a dataset with K classes.
- $f_i = \frac{observation\ number\ of\ class\ i}{observation\ number\ in\ the\ dataset}$ the frequencies of each labels with $f_1 \geq f_2 \geq ... \geq f_k$.
- t the percentage of desired observations in the under-represented class.

The first rebalancing technique is to create sub datasets with the same frequency of each class.

We define $\tilde{f}_i = \frac{f_k * t}{f_i}$ the percentage to be drawn of each label.

Randomly Balanced

The second rebalancing technique is to have datasets which frequencies will be different for each neural network.

Let :

- $\tilde{f}_i$ define as before
- a such that $a + t \leq 1$
- $\vec{U}$ a vector of independent variable uniformly distributed on $[-a, a]$

We define $\ddot{f}_i = \tilde{f}_i + U_i$ the percentage to be drawn of each label.

Lightly Balanced

Under sampling the major class such as the minor class account for 10% of our final data set.

- Distribution close to the original

- Distribution which can help us learn our minority class

Summary

We compared different methods to perform the embedding :

rand: All the words are randomly initialized and then modified during training.

static: The embedding network is initialized using Fasttext.

non-static: Same as static but word vectors are fine-tuned.

Categories	min	mean	var	median	max
Standard claims (uncensored)	0	1	1	0,75	16.3
Extreme claims (uncensored)	0,25	3.83	6,93	3,08	16.3
Standard claims (after KM)	0	1.25	2.26	0,83	16.3
Extreme claims (after KM)	0,25	5.24	11.7	4.17	16.3

Table: Empirical statistics on the variable T, before and after correction by Kaplan-Meier weights ("after KM"). The category "Extreme claims" corresponds to the situation where $I = 1$ for $x = 3\%$ of the claims, while "Standard claims" refers to the 97% lower part of the distribution of the final amount.

Rank	Extreme	Normal
1	insurer 90%	insurer 87%
2	third party 56%	third party 61%
3	injured 38%	front 46%
4	to ram 30%	way 41%
5	to hit 24%	backside 40%
6	motorcycle 18%	left 20%
7	driver 17%	right 18%
8	pedestrian 16%	side 17%
9	inverse 15%	to shock 14%
10	deceased 13%	control 10%

Table: Ranking of the words (translated from French) used in the reports, depending on the category of claims (Extreme corresponds to $I = 1$ and Standard to $I = 0$.)

On minority class

Method	Model	type Embedding	precision	recall	f1-score
Classical	Expert		0.94	0.05	0.02
	Random Forest	static	0.20	0.22	0.21
	Gradient Boosting	static	0.17	0.31	0.22
	CNN	non-static	0.78	0.06	0.12
	LSTM	non-static	0.66	0.11	0.19
Balanced	CNN	non-static	0.28	0.48	0.33
	LSTM	non-static	0.28	0.46	0.35
Randomly	CNN	non-static	0.33	0.42	0.37
	LSTM	non-static	0.34	0.48	0.40
Lightly	CNN	non-static	0.41	0.44	0.42
	LSTM	non-static	0.47	0.40	0.43

Acknowledgments

Thanks to Olivier Lopez (Sorbonne Université, Paris, France), Yann Mercuzot (Pacifica, Paris, France).

Thank you for listening

mail: isaaccohensabban@gmail.com

Linkedin : https://www.linkedin.com/in/isaaccohensabban/