Classifying unstructured text
Deterministic and machine learning approaches

“Using a simple tool to solve a complex problem does not result in a simple solution.” Larry Wall

Stephanie Fischer
Dr. Christian Winkler

Apache Big Data Sevilla, 15th November 2016
Agenda

01 About us
02 Text statistics
03 Categories
04 Text classification
05 Conclusion and outlook

Speaker

Stephanie Fischer
Product Owner Text Analytics
Big Data, Agile & Change
mgm consulting partners

Dr. Christian Winkler
Enterprise Architect
Big Data, Data Science
mgm technology partners
01

About us
Stephanie and Christian according to their browser history
02

Text statistics
Comparing *word frequency* of news from Reuters, Telegraph, Aljazeera

Reuters World News

- # 163,919 headlines
- 🕒 9 years

Telegraph

- # 958,996 headlines
- 🕒 9.5 years

Aljazeera

- # 94,309 headlines
- 🕒 8.5 years

Visualizations created with Apache Solr and D3.js, see our talk from Apache Big Data Vancouver 2016 here: [https://bigdata.mgm-tp.com/apache/](https://bigdata.mgm-tp.com/apache/)
03 Categories
Finding meaningful categories. Each text is different. Challenge accepted!
Comparing pre-defined categories of Al Jazeera, Reuters...
... and the Telegraph categories
It's not so easy.
Our selection: Functionally relevant, mutually exclusive categories derived from Telegraph categories
Finding **meaningful categories** for the Telegraph News was fun!

Let's go on and do a whole **text classification experiment**. Our **aim** is to **classify 1 million Telegraph News documents** with an ML algorithm.

While doing this we want to **find out**...

... if a **ML algorithm** will be able to classify the Telegraph news documents

... what are the **steps** we need to work out in order to make the ML algorithm work?

Handy for us: We will be able to **train** the **ML algorithm with the pre-classified data set** of the Telegraph News!
04
Text classification
Typical text classification projects and our experiment set-up

Typical set-up: no classification scheme, no classified data

- Choose data to be classified
- Manually classify chosen data set
- Train ML algorithm with classified data set
- Apply trained ML algorithm to complete data set
- Manual QA data set samples

Our Telegraph experiment with pre-classified documents

- Choose data to be classified
- Get already existing classifications for chosen data
- Train ML algorithm with classified data set
- Apply trained ML algorithm to complete data set
- Automatic QA complete data set

Advantages for us:

- No manual classification & QA necessary
- Existing classification scheme
- Playground easily set up
- Free to choose both manual data set & categories
Our experiment for the next 30 minutes

Typical set-up: no classification scheme, no classified data

Choose data to be classified
Manually classify chosen data set
Train ML algorithm with classified data set
Apply trained ML algorithm to complete data set
Manual QA data set samples

Our Telegraph experiment with pre-classified documents

Choose data to be classified
Get already existing classifications for chosen data
Train ML algorithm with classified data set
Apply trained ML algorithm to complete data set
Automatic QA complete data set

Our aims in the next 30 minutes:

- Train & apply the ML algorithm to 1 of Telegraph News
- See how well ML performs
This process sounds easy and very structured. The people in the audience who have already done text classification projects probably now that in reality, data can become pretty challenging.

The next slides show you the process of how we classified 1 Million Telegraph news.

What is the reality we deal with?

And what are good practices/our learnings?
Getting started: Preparing data for and executing ML

1. Choose data to be classified
   - Naïve approach
     - Random selection

2. Manually classify data set
   - Easy to classify
     - But expensive

3. Apply ML algorithm
4. Get classifications

5. Manual QA
6. Measure results
The result is **BAD!**

**WHY?**

Let's take a step back and find out:
How does **ML WORK?**
How can I **MEASURE** its results?
ML algorithm explained – Support Vector Machine (SVM)

Machine learning is linear algebra
- Need to discretize first
Categories are already discrete
More complicated for text
  - Bag of words = detect words
  - TF/IDF matrix = use document and total frequency
Many different possible learning models
- Support Vector Machines (most popular)
- Neural Network
- Random forest
- Decision tree
Preparation of manually classified set

Choosing set for manual classification

- Select documents with highest word variability
  - Metric: Word heterogeneity = Number of words in all documents (→ stopwords)
  - Even distribution
  - Long tail distribution (→ many, many words use infrequently)
- Complicated: knapsack-like problem
- Use an approximate approach (like genetic algorithm)
- Crucial for all following tasks

1. Good situation:
The manually classified data set contains all the words of the complete data set.

<table>
<thead>
<tr>
<th>Word heterogeneity in manual set</th>
<th>Word heterogeneity complete data set</th>
</tr>
</thead>
<tbody>
<tr>
<td>w01    w02    w03</td>
<td>w01    w02    w03</td>
</tr>
<tr>
<td>w04    w05    w06</td>
<td>w04    w05    w06</td>
</tr>
<tr>
<td>w07    w08    w09</td>
<td>w07    w08    w09</td>
</tr>
<tr>
<td>w10    w11    w12</td>
<td>w10    w11    w12</td>
</tr>
<tr>
<td>w13    w14    w15</td>
<td>w13    w14    w15</td>
</tr>
<tr>
<td>w16    w17    w18</td>
<td>w16    w17    w18</td>
</tr>
</tbody>
</table>

2. Not so good situation:
The manually classified data set contains only a fraction of all the words in the complete data set.

<table>
<thead>
<tr>
<th>Word heterogeneity in manual set</th>
<th>Word heterogeneity complete data set</th>
</tr>
</thead>
<tbody>
<tr>
<td>w19    w20    w21</td>
<td>w19    w20    w21</td>
</tr>
<tr>
<td>w22    w23    w24</td>
<td>w22    w23    w24</td>
</tr>
<tr>
<td>w25    w26    w27</td>
<td>w25    w26    w27</td>
</tr>
<tr>
<td>w28    w29    w30</td>
<td>w28    w29    w30</td>
</tr>
<tr>
<td>w31    w32    w33</td>
<td>w31    w32    w33</td>
</tr>
<tr>
<td>...    ...    ...</td>
<td>...    ...    ...</td>
</tr>
<tr>
<td>w99</td>
<td></td>
</tr>
</tbody>
</table>
Intelligently choose data set to be classified manually

- Choose training data set in a way to create maximal word overlap with complete data set
  - \( W_M = \{ \text{words in training set} \} \)
  - \( W_C = \{ \text{words in complete set} \} \)
  - find maximum for \( |W_C \cap W_M| = |W_M| \)
- Improved approach: choose training set to minimize headlines with unknown words in complete data set
  - Find minimum for \( |C \cap \overline{W_M}| \)
  - More complicated, but worth it

- Optimize for high variability and high usage
Measure classification quality: precision and recall

**Precision**
- “positive predictive value”
- Precision is the probability that a (randomly selected) retrieved document is classified correctly

**Recall**
- Sensitivity or “true positive rate”
- Recall is the probability that a (randomly selected) classified document is found

**Example**
- Africa has very high precision for category “Africa”, but bad sensitivity (recall)

[Diagram showing precision (P) and recall (R) with arrows and overlapping circles]
Now we know why the naive approach of preparing data for and executing ML is not enough. Let’s try the following instead…

Choose data to be classified
- Calculate text metrics
- Define goals
- Use optimization

Manually classify data set
- Easy to create
- But expensive
- Optimal set

Apply ML algorithm
- Training and test data set
- Calculate Precision + Recall
- Crossfolding, use different algorithms

Get classifications
- Better results

Attention:
ML remembers words
→ It can only classify text with known words

Necessary steps to successfully apply ML
- Measure quality of ML
- Training set and test set
- Precision and recall
- Apply ML to whole data set
- Manual QA

Manual QA
Optimal set
Easy to create
But expensive
Training and test data set
Calculate Precision + Recall
Crossfolding, use different algorithms
Better results

Necessary steps to successfully apply ML
- Measure quality of ML
- Training set and test set
- Precision and recall
- Apply ML to whole data set
- Manual QA
What we have done & achieved so far

1. Data cleaning and preparation: Docs with same headline but different classification removed

2. Category definition: Mutually exclusive, functionally relevant

3. Precision & Recall per categories and different training/test-sets

4. Eliminating Longtail

- abdulrahim kerimbakiev
- abbot placid spearritt
- Abdullahi Sudi Spearritt
- abdulah Alhamiri
- abib sarajuddin
- acer nethercott
- abdulli feghoul
The result is **BETTER!**

Now...

... what **options** do you have if you don’t have a pre-categorized data set **to train your ML**?

... or your **manually classified data set is too small**?
What to do about the things that can still go wrong

Manually classified data set is too small for training
- Data set is too heterogenous
- ML cannot detect patterns
- Bad precision and recall

Extend data set
- Requires manual classification
- Too expensive

Try to understand structure of manual classification
- Find category-specific keywords
- Find patterns
- Use NLP etc.

→ Extension of training set by deterministic classification
Improved approach with deterministic extension

1. Choose data set to be classified
2. Calculate text metrics
3. Define goals
4. Use optimization
5. Manually classify data set
6. Extend data set
7. Rule engine with deterministic rules
8. Pattern matching
9. Get classifications
10. Larger training and test set
11. Calculate Precision + Recall
12. Use different algorithms
13. Apply ML algorithm
14. Manual QA

- Easy to create
- But expensive
- Optimal set
- Even better results
Iterate and Improve

Be prepared for a long journey: Often results get better incrementally

1. Find/adjust Categories

<table>
<thead>
<tr>
<th>Task</th>
<th>Tools</th>
</tr>
</thead>
<tbody>
<tr>
<td>Visualization</td>
<td>Solr &amp; D3</td>
</tr>
<tr>
<td>Clustering</td>
<td>Solr</td>
</tr>
<tr>
<td>Brain &amp; Expertise</td>
<td>Data analytics</td>
</tr>
</tbody>
</table>

2. Documents to assign categories manually

<table>
<thead>
<tr>
<th>Task</th>
<th>Tools</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manual classification</td>
<td>Mechanical turk</td>
</tr>
<tr>
<td>Deterministic classification</td>
<td>Rule engine</td>
</tr>
</tbody>
</table>

3. Train and apply ML

<table>
<thead>
<tr>
<th>Task</th>
<th>Tools</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training</td>
<td>R, Mahout, Spark</td>
</tr>
<tr>
<td>Test</td>
<td>R, Mahout, Spark</td>
</tr>
</tbody>
</table>

4. Measure results

<table>
<thead>
<tr>
<th>Task</th>
<th>Tools</th>
</tr>
</thead>
<tbody>
<tr>
<td>Check precision, recall, f-measure</td>
<td>R</td>
</tr>
<tr>
<td></td>
<td>Apache Mahout</td>
</tr>
<tr>
<td></td>
<td>Spark ML</td>
</tr>
</tbody>
</table>

Output: Categorized Data

+ Histograms
+ Visualization
+ Metrics
+ Category specific keywords
+ Hierarchies, rules, entities

Input: Categories + Data
Talking about longtail: Variability

Reasons for longtail
- Flat longtail via dictionary-type texts
- Decreasing longtail from domain specific language

Analyze the longtail
- Count words
- Measure heterogenity

Elimination strategies
- Foreign language detection
- Eliminate typos (n-grams)
- Manual classification if not too many documents
- Put into separate category (aka „miscellaneous“)
How to make the decision in your data analytics project data-driven? ... measureable?
Metrics help making objective decisions during the project

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Docs</td>
<td>Unique docs</td>
<td>Unique docs</td>
<td>Classified docs</td>
<td>Precision</td>
</tr>
<tr>
<td>Unique docs</td>
<td>Unique docs</td>
<td>Unique docs</td>
<td>Long tail</td>
<td>Recall</td>
</tr>
<tr>
<td>Unique words</td>
<td>Unique words</td>
<td>Unique words</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Categories</td>
<td>Categories</td>
<td>Categories</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The project is finished when your cost/benefit ratio (or its prediction) of classifying the longtail becomes negative.
Conclusion and outlook
10 Lessons learned

<table>
<thead>
<tr>
<th>Really naïve</th>
<th>Sounds clever</th>
<th>Sounds naïve</th>
<th>Really clever</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run it on your notebook</td>
<td>Complex data structure &amp; complicated classification scheme</td>
<td>Trying to understand ML</td>
<td>Thinking the functional specification is finished before the project is finished</td>
</tr>
<tr>
<td>Increase the ML test &amp; training set manually and deterministically</td>
<td>Check data heterogenity immediately, then choose technic</td>
<td>Design manually classified data set very simple so ML will reach a high Precision/Recall</td>
<td>Get creative to find useful pre-categorized data</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Mutually exclusive categories</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Understand data qualitatively &amp; quantitatively</td>
</tr>
</tbody>
</table>

Some text for this page

There are two preconditions to functional classification:
Categories are well known, someone else has specified it totally great, überschneidungsfrei. OR there is a training data set big enough for your data set, already pre-classified → You don't need to classify yourself. BUT ATTENTION: You also need to get to know this data set and the logic behind it really well. If the logic is bad, insufficient, nicht überschneidungsfrei, ... you should not use it. Therefore you always need to do manual QA.

What more pre-classified data set are there?

If there are no well-known categories with good-enough quality already existing, you need to bite the bullet and create this probably pretty innovative, very data-set specific categorization scheme yourself. This specification will be finished only when the project is finished.

Manual classification especially important

You need to deeply get involved with the data set.

Process: Feels chaotic, but the only way out is through.

Trying to understand ML

Skala nochmal checken

Run it on your notebook
Getting more pre-categorized data by

- Categories from other sources
- Semantic extraction
- NLP
- Meaning

Not yet analyzed text is everywhere

- Discretization helps in understanding
- Toolbox with ML. Deterministic rules helpful

Outlook

Big potential: use already classified data to classify new data
Innovation Implemented.

**mgm technology partners GmbH**
Frankfurter Ring 105a
80807 München
Tel.: +49 (89) 35 86 80-0
Fax: +49 (89) 35 86 80-288
[http://www.mgm-tp.com](http://www.mgm-tp.com)

**mgm consulting partners GmbH**
Holländischer Brook 2
20457 Hamburg
Tel.: +49 (0) 40 / 80 81 28 20-0
Fax: +49 (0) 40 / 80 81 28 20-388
[http://www.mgm-cp.com](http://www.mgm-cp.com)