# Crude oil prices predictions using NLP

Maxime Genin, Arthur Morlot, Sophie Trastour

## Problem

In our project, we focused on the correlations that could exist between the commodity prices of crude oil and external factors highlighted in newspapers. Thanks to machine learning and NLP (natural language processing) techniques, more and more documents can be processed in a semi-automated way. We used topic modeling (Latent Dirichlet Allocation) to extract the main topics from the articles in newspapers such as New-York Times, Reuters and the Associated Press so as to predict the movement of the stock oil price.

## Data

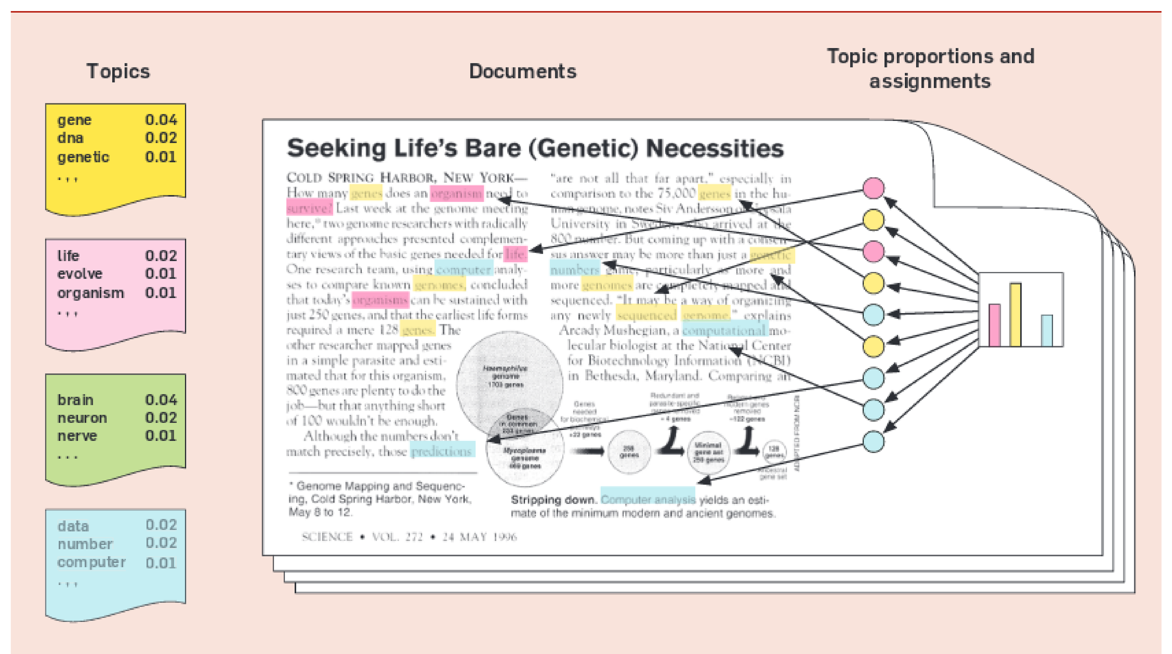
We used the NY Times API to get automatically all articles from 1986 to 2015 containing the words “oil price", which corresponds to around 30,000 articles. We were able to extract the headline, an abstract and a snippet. We used several API, the nytimesarticle package, and the python function time.sleep() to avoid the limitations of 5 articles per second and 1,000 articles per day. We used the stock oil price data from the EIA website. Finally, in order to improve the quality of our predictions, we adjusted the stock oil prices with the US inflation from 1974.

## NLP method: Latent Dirichlet Allocation (LDA)

We decided to use a LDA technique because of its capacity to capture multiple topics within a document (a complete description can be found in [1]). The intuition behind the LDA is the following: each document is a “mixture" of different topics. A document may be 90 % about “oil" and 10% about “cars" (see figure 1). We now make an assumption: if a document is composed of 90 % about “oil" and 10% about “cars", then it is constructed by randomly sampling 90 % of its words from a distribution about “oil", and 10% from a distribution about “cars" (the ordering of the words does not matter to the algorithm). We end up with three hidden variables to explain our corpus:

- The topics, that is to say the words distribution inside a topic
- Per-document topic distributions
- Per-document per-word topic assignments

Each of them can be set to a specific prior before we run the algorithm, to encode information known by humans about the subject.



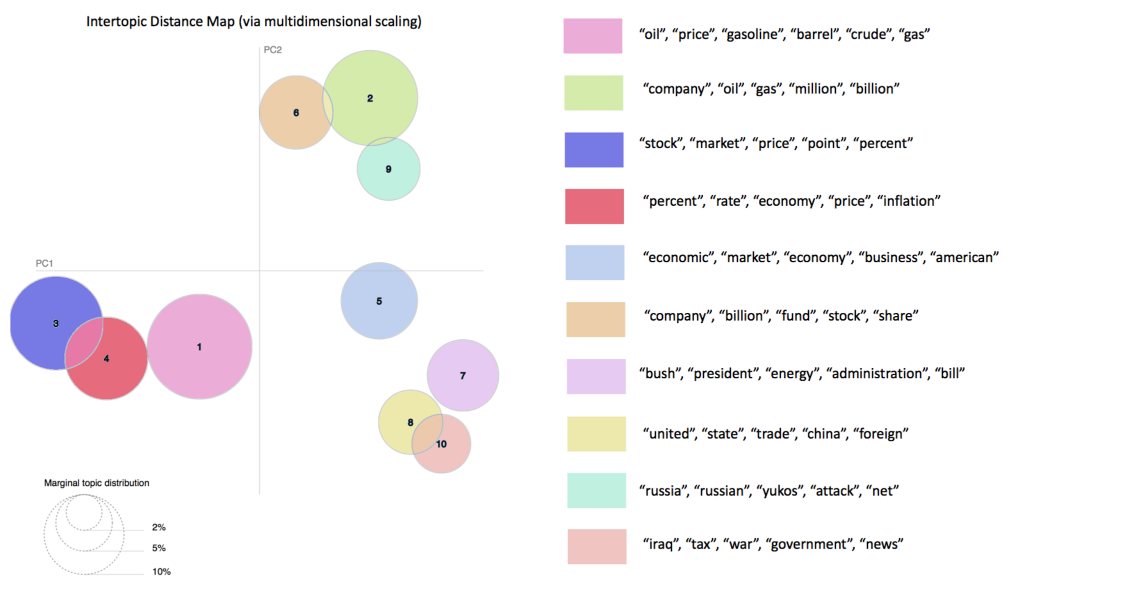

The LDA then uses inference algorithms to compute the posterior on these distributions and infer the more likely ones.

## Preprocessing

There are packages available to do topic modeling in python like gensim and pyLDAvis [3]. We preprocessed the newspapers articles by:

- setting to lower case
- removing the punctuation
- removing stop words (like “I", “my", “their") which does not carry meaning
- lemmatizing the words (so that “like" and “liked" would be treated the same way)
- removing the numbers

The list of stop words and the data for lemmatizing words were extracted from the nltk package in python. We first set the number of topics to 10. The topics found were:




We extracted the topics distributions of each article, and selected the “main" topic of this latter. We then merged all the articles from the same day together and ended up with the following dataset: for each day, we had the number of articles published, and their topic distribution.

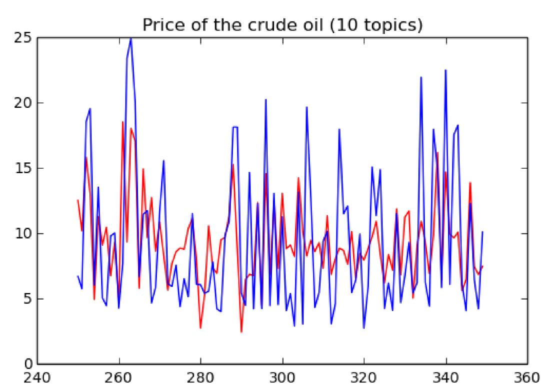
## Results

Linear regression with 10 topics:

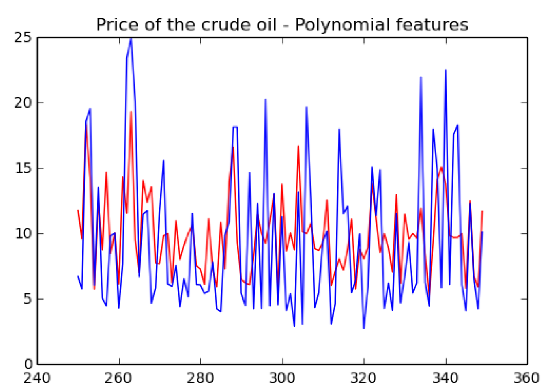
| Features | Outputs | Comments |
|---|---|---|
| Daily proportions for each topic | Daily returns | Too volatile |
| Daily proportions for each topic | Daily prices | Better but still volatile |
| Monthly proportions for each topic | Monthly returns | Too volatile |
| Monthly proportions for each topic | Monthly prices | Reasonable results |

So as to select the best model, we used a 10-fold cross validation strategy adjusting the following parameters:

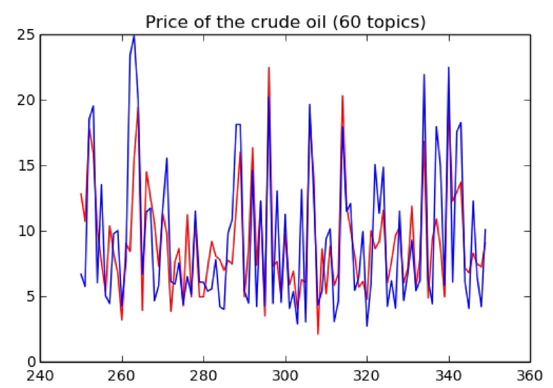
- Number of topics: 10, 40, 60 and 100
- Polynomial features to introduce non-linearity
- Different LDA models (stopwords list, topics)
- Type of regression: Theil-Sen, least-square, ridged least-square, locally weighted regression



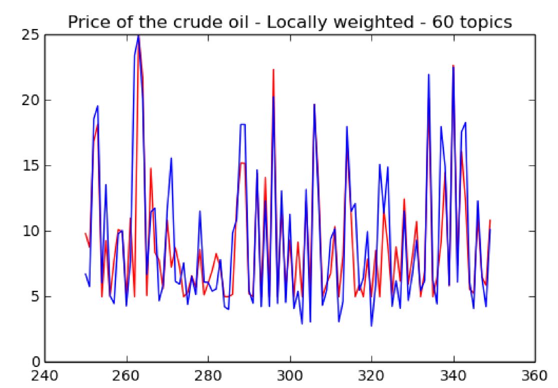

Linear regression ($R^2 = 0.46$, 10 topics)




Polynomial features ($R^2 = 0.42$, 10 topics)

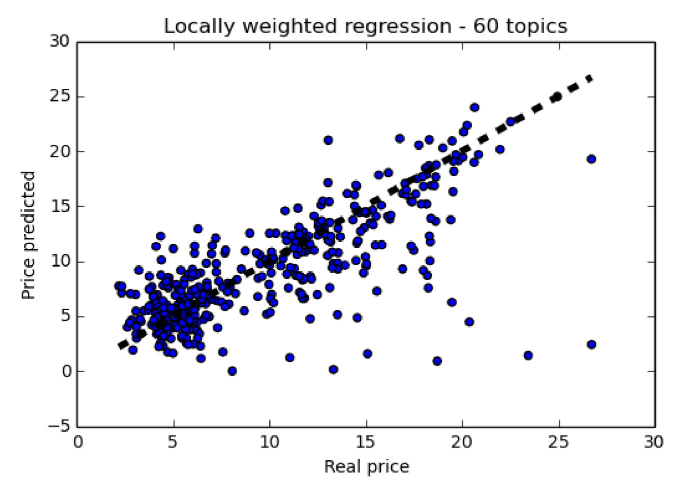



Ridged least-square ($R^2 = 0.54$, 60 topics)




Locally weighted ($R^2 = 0.70$, 60 topics)

**Best model found:** locally weighted regression with 60 topics and no polynomial features ($R^2 \approx 0.70$)




## Future works

There are several ways to improve our model:

- Add new features: topics distribution for oil and gas companies (Shell, Exxon, Chevron, ...), trend of the market
- Implement the model in an online fashion: determine the update frequency of the model according to new press releases
- Combine the model with a trading strategy using reinforcement learning

## References

[1] Blei, David M. Probabilistic topic models. Communications of the ACM, 2012, vol. 55, no 4, p. 77-84.

[2] Newman, D. J., & Block, S. (2006). Probabilistic topic decomposition of an eighteenth-century American newspaper. Journal of the American Society for Information Science and Technology, 57(6), 753-767.

[3] Sievert, C., and Shirley, K.E., 2014, LDAvis: A Method for Visualizing and Interpreting Topics: Proceedings of the Workshop on Interactive Language Learning, Visualization, and Interfaces, p. 63-70