Comparison of Neural Network Architectures for Sentiment Analysis of Russian Tweets

Speaker: Konstantin Arkhipenko[1,2] (arkhipenko@ispras.ru)

Ilya Kozlov[1,3] Julia Trofimovich[1]

Kirill Skorniakov[1,3] Andrey Gomzin[1,2] Denis Turdakov[1,2,4]

[1]Institute for System Programming of RAS, Moscow, Russia
[2]Lomonosov Moscow State University, CMC faculty, Moscow, Russia
[3]MIPT, Dolgoprudny, Russia
[4]FCS NRU HSE, Moscow, Russia

June 2, 2016

Contents

Contents

SentiRuEval-2016: The task

- Object-oriented sentiment analysis of Russian tweets
- Given a tweet t_i and set $O_{t_i} \subset O$ of objects mentioned in t_i, **for each** $o \in O_{t_i}$:
  - mark t_i as negative, neutral or positive towards o

SentiRuEval-2016: The data

- Two domains: **banks** and **telecommunication companies**
- Train:
  - Banks: 9392 tweets
  - TC: 8643 tweets
- Test:
  - Banks: 19586 tweets (3313 of them used for evaluation)
  - TC: 19673 tweets (2247 of them used for evaluation)
- Imbalanced: 65% of tweets in train data are neutral

SentiRuEval-2016: The metrics

- Precision:

$$precision = \frac{truePositiveMarks}{truePositiveMarks + falsePositiveMarks}$$

- Recall:

$$recall = \frac{truePositiveMarks}{truePositiveMarks + falseNegativeMarks}$$

- F1-score:

$$f1 = \frac{2 \times precision \times recall}{precision + recall}$$

- **F1-score, macro-averaged over negative and positive classes; used for evaluation:**

$$f1Macro = 0.5 \times (f1_{negative} + f1_{positive})$$

Contents

Our solutions

- We focused on determining overall sentiment of the whole tweets
- Given a tweet t_i and set $O_{t_i} \subset O$ of objects mentioned in t_i, determine sentiment $s_{t_i} \in \{negative, neutral, positive\}$ of t_i and **for each** $o \in O_{t_i}$:
  - mark t_i as s_{t_i} towards o

Why neural networks?

- Modern NN architectures (e.g. recurrent neural networks) achieve state-of-the-art in many NLP problems, outperforming shallow machine learning approaches
- Lots of powerful, efficient, easy-to-use deep learning libraries evolved over last few years

Word embeddings: word2vec

- Introduced by Tomas Mikolov (now at Facebook AI Research)
- Maps words into vector space
- Based on simple feed-forward neural network
- Captures syntactic and semantic regularities
- Helps to overcome data sparsity in our task

Word embeddings: word2vec

- We trained word2vec on 3.3 GB of Web users' comments from:
  - ВКонтакте (`https://vk.com/`)
  - Эхо Москвы (`http://echo.msk.ru/`)
  - Свободная Пресса (`http://svpressa.ru/`)
- The following parameters were used:
  - Continuous Bag-of-Words architecture
  - 10 negative samples for every prediction
  - word embeddings' dimensionality of 200
  - 5 training iterations over corpus

Baseline: SVM + domain adaptation

- For every tweet:
  - convert it to sequence of corresponding word2vec word embeddings. Punctuation and words that are not in word2vec vocabulary are discarded
  - form a **tweet embedding** by averaging vectors in this sequence and feed it into support vector machine (SVM) classifier
- Domain adaptation:
  - we discovered that source domain (train data) and target domain (test data) are drawn from different probability distrubutions
  - **sample reweighting:** give higher weights to samples that look like target samples and don't look like source samples

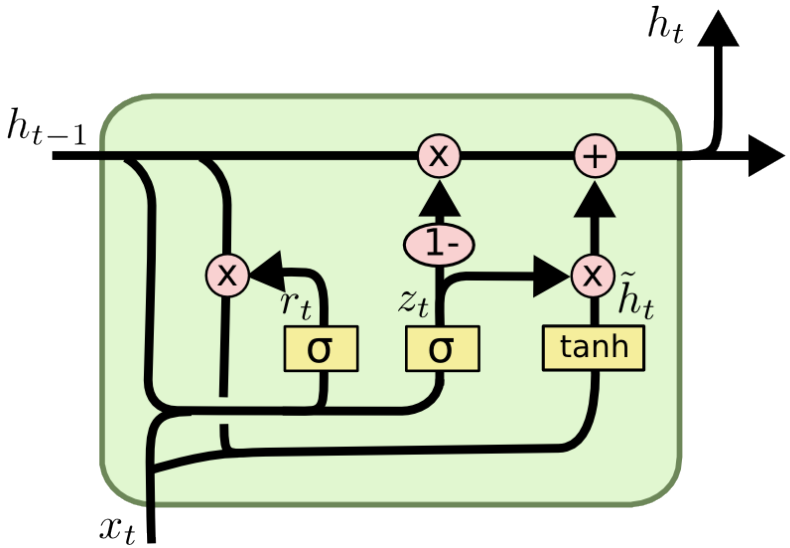
CNN-based solution

- For every tweet:
  - form a tweet embedding
  - also form an **additional tweet embedding** by getting element-wise maximum of all word embeddings in the sequence
  - concatenate these tweet embeddings and feed the result into convolutional neural network (CNN)
- Convolutional neural network:
  - convolutional layer with 8 kernels of width 10
  - dense layer: 3 neurons with softmax activation that predict probabilities of each class (negative, neutral and positive)
  - 10 training epochs
- **Yes, this solution is quite silly... ☺**

  (but not all possible CNN-based approaches)

Recurrent neural networks: Gated Recurrent Unit (GRU)

- RNNs are suitable for processing sequence data



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

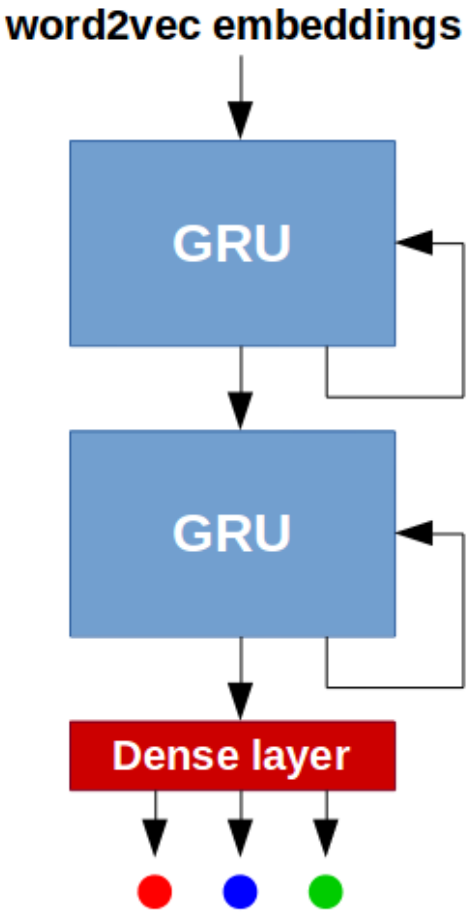
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

(http://colah.github.io/posts/2015-08-Understanding-LSTMs/)

RNN-based solution

- Neural network takes sequence of word2vec embeddings of words in the tweet as an input
- NN architecture:
  - two GRU cells with input/output dimensionality of 200; dropout is applied to the output of second cell
  - dense layer: 3 neurons with softmax activation that predict probabilities of each class
- Implemented using Keras library:
  - `http://keras.io/`
  - only 200 lines of code
- 20 training epochs, batch size of 8



Evaluation results: Macro F1-score

Solution	Banks domain / Rank	TC domain / Rank
CNN	0.4832 / 21st	0.4704 / 41st
GRU	**0.5517 / 1st**	**0.5594 / 1st**
Ensemble*	0.5352 / 2nd	0.5403 / 9th

*combination of CNN, GRU, and Baseline (SVM + domain adaptation) solutions

Contents

Conclusion and future work

- Our CNN-based solution is very silly
- We are not deep learning experts (yet)
- We had little time for competition
- We did not use any lexicons and performed very little preprocessing
- We did not explore hyperparameter values properly
- **However, we have won the competition ☺**
- Next year we are going to improve the results significantly:
  - discover optimal NN architectures and find better hyperparameters
  - use domain adaptation in neural networks
  - ...

Thank you!

Questions?