Keyword extraction with semantic surprisal from LDA and LSA: a comparison of topic models

J. Nathanael Philipp[1,2] Max Kölbl[3] Yuki Kyogoku[2] Tariq Yousef[2] Michael Richter[2]

[1]Sorbisches Institut [2]Universität Leipzig [3]Osaka University

Aim

A study on keyword extraction in German, comparing the performance of Latent Dirichlet Allocation (LDA) and Latent Semantic Analysis (LSA) as part of the Topic Context Model (TCM).
The comparison of LDA and LSA proceeds in two steps: (i) the quality of the words' surprisal values for keyword extraction is directly compared, (ii) the surprisal values are input to a Recurrent Neural Network (RNN), which predicts keywords.

Topic Context Modell

The Topic Context Model (TCM) calculates the information-theoretical measure surprisal as a context-based feature of words. Surprisal is a contextualised information measure based on conditional probabilities. TCM evaluates the topic- and topic-word distributions in a text for each word.

$$surprisal(w) = -\log_2 P(w|\text{CONTEXT})$$

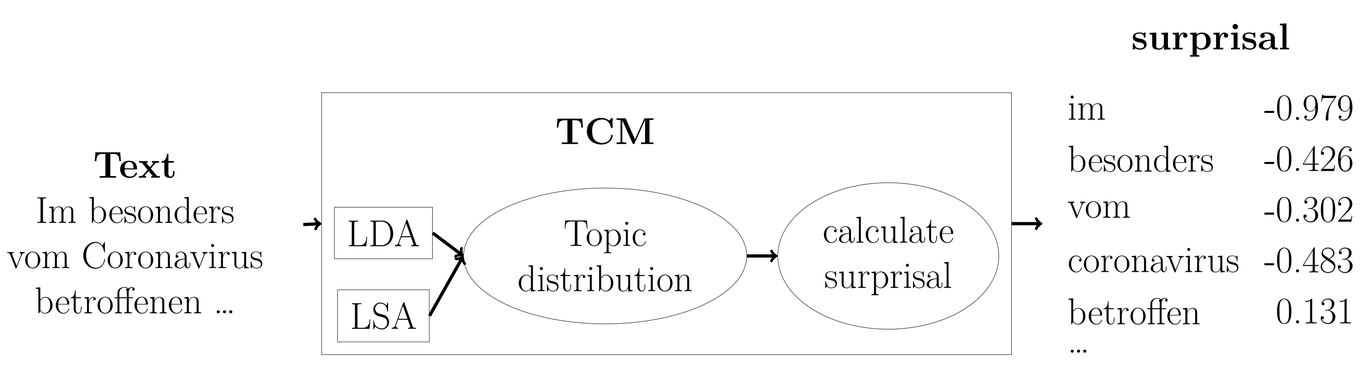

Figure: Illustration of the Topic Context Model.

Scoring function for the surprisal values, used for the input to the a neural network.

$$\widehat{surprisal}(w_d) = \tanh \frac{surprisal(w_d) - \mu_d}{\sqrt{\sigma_d^2 + \epsilon}}$$

Latent Semantic Analyis

Latent Semantic Analyis (LSA) reduces the dimension of a text using *Singular Value Decomposition* (SVD). For a document d with topic vector v_d, we compute the likelihood $P(w_d)$ of a word w_d to appear in d by checking its appearance in other texts, where $\delta(w_d, s)$ is 1 if w_d appears in s and 0 otherwise.

$$surprisal(w_d) = -\log_2 P(w_d|d)$$
$$P(w_d) = \frac{\sum_{s \in M} \delta(w_d, s)\,(1 + \cos\langle v_s, v_d\rangle)}{\sum_{s \in M} 1 + \cos\langle v_s, v_d\rangle}$$

Latent Dirichlet Allocation

LDA is a statistical topic model that tries through a generative process to detect and identify the topics that appear in a document and which words belong to them.

$$surprisal(w_d) = \overline{surprisal}(w_d) = -\frac{1}{n} \sum_{i=1}^{n} \log_2 P(w_d|t_i)$$
$$P(w_d|t_i) = \frac{c_d(w_d)}{|d|} WT_{w_d,t_i} P(t_i|d)$$

Recurrent Neural Network

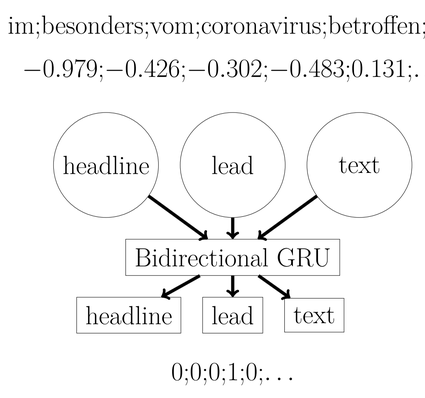

Figure: Schematic of the RNN architectures with text and $\widehat{surprisal}$ as input.

Results

Model	a1	a2	a3	a4	a5	Precision	Recall	F1
Baselines								
NER	**0.8771**	0.4528	0.1896	0.0918	0.0558	0.1249	0.3808	0.1784
TextRank_5	0.7781	0.3633	0.1998	0.1724	0.1707	0.2161	0.4341	0.2725
TextRank_{10}	0.8606	**0.5112**	**0.3078**	0.2550	0.2398	0.1371	0.5351	0.2094
$\text{RNN}_{T,256}$	0.6902	0.3636	0.2919	0.2847	0.2840	0.5363	0.4669	0.4638
$\text{RNN}_{T,512}$	0.6989	0.3795	0.3012	0.2936	0.2926	0.5544	0.4772	0.4773
$\text{RNN}_{T,1024}$	0.6929	0.3702	0.2926	0.2853	0.2847	0.5562	0.4697	0.4759
$\text{RNN}_{T,2\times256}$	0.6764	0.3557	0.2734	0.2678	0.2672	0.5416	0.4531	0.4622
$\text{RNN}_{T,2\times512}$	0.6635	0.3504	0.2751	0.2698	0.2692	**0.5615**	0.4471	0.4660
Topic Context Models with LDA and LSA								
LDA_5	0.0281	0.0043	0.0036	0.0033	0.0033	0.0059	0.0126	0.0076
LDA_{10}	0.0888	0.0185	0.0135	0.0135	0.0132	0.0096	0.0427	0.0152
$\text{LDA}_{10\%}$	0.1219	0.0228	0.0162	0.0162	0.0159	0.0107	0.0552	0.0173
$\text{LDA}_{20\%}$	0.2665	0.0816	0.0532	0.0479	0.0476	0.0128	0.1371	0.0228
$\text{LDA}_{30\%}$	0.3653	0.1374	0.0869	0.0766	0.0753	0.0124	0.2018	0.0230
LSA_5	0.0535	0.0149	0.0132	0.0129	0.0129	0.0113	0.0290	0.0154
LSA_{10}	0.1433	0.0469	0.0370	0.0363	0.0363	0.0160	0.0799	0.0256
$\text{LSA}_{10\%}$	0.1780	0.0528	0.0376	0.0367	0.0363	0.0171	0.0948	0.0279
$\text{LSA}_{20\%}$	0.4326	0.2057	0.1526	0.1410	0.1404	0.0228	0.2725	0.0411
$\text{LSA}_{30\%}$	0.6380	0.3785	0.2949	0.2711	0.2682	0.0246	0.4465	0.0459
Recurrent Neural Networks with LDA								
$\text{RNN}_{LDA,256}$	0.3362	0.1159	0.1143	0.1143	0.1143	0.3249	0.2038	0.2355
$\text{RNN}_{T,LDA,256}$	0.7028	0.3768	0.3015	0.2933	0.2929	0.5571	0.4785	0.4798
$\text{RNN}_{LDA,512}$	0.2645	0.0908	0.0888	0.0888	0.0888	0.2518	0.1605	0.1845
$\text{RNN}_{T,LDA,512}$	0.7028	0.3768	0.3048	**0.2959**	**0.2952**	0.5576	0.4792	0.4812
$\text{RNN}_{LDA,1024}$	0.3445	0.1159	0.1133	0.1133	0.1133	0.3245	0.2069	0.2371
$\text{RNN}_{T,LDA,1024}$	0.6767	0.3639	0.2860	0.2807	0.2801	0.5423	0.4602	0.4645
$\text{RNN}_{LDA,2\times256}$	0.3554	0.1219	0.1189	0.1189	0.1189	0.3349	0.2144	0.2455
$\text{RNN}_{T,LDA,2\times256}$	0.6988	0.3791	0.3025	0.2932	0.2926	0.5586	0.4775	**0.4817**
$\text{RNN}_{LDA,2\times512}$	0.3326	0.1169	0.1139	0.1139	0.1139	0.2299	0.3094	0.2031
$\text{RNN}_{T,LDA,2\times512}$	0.6929	0.3686	0.2933	0.2853	0.2847	0.5527	0.4696	0.4751
Recurrent Neural Networks with LSA								
$\text{RNN}_{LSA,256}$	0.2503	0.0803	0.0803	0.0803	0.0803	0.25	0.1491	0.1767
$\text{RNN}_{T,LSA,256}$	0.6823	0.3613	0.2870	0.2810	0.2804	0.5461	0.4620	0.4680
$\text{RNN}_{LSA,512}$	0.2345	0.0836	0.0822	0.0822	0.0822	0.2282	0.1446	0.1669
$\text{RNN}_{T,LSA,512}$	0.6879	0.3771	0.2966	0.2906	0.2900	0.5510	0.4708	0.4745
$\text{RNN}_{LSA,1024}$	0.4128	0.1404	0.1328	0.1328	0.1328	0.2774	0.3722	0.2472
$\text{RNN}_{T,LSA,1024}$	0.6952	0.3748	0.2962	0.2900	0.2890	0.4739	**0.5483**	0.4731
$\text{RNN}_{LSA,2\times256}$	0.2490	0.0925	0.0908	0.0908	0.0908	0.2389	0.1552	0.1768
$\text{RNN}_{T,LSA,2\times256}$	0.6863	0.3616	0.2906	0.2837	0.2834	0.5591	0.4638	0.4721
$\text{RNN}_{LSA,2\times512}$	0.1156	0.0423	0.0423	0.0423	0.0423	0.1151	0.0723	0.0842
$\text{RNN}_{T,LSA,2\times512}$	0.7034	0.3699	0.2949	0.2867	0.2863	0.5313	0.4756	0.4671

Table: Precision, recall, F1-measure, and the accuracy-values (a1 – a5) of the employed methods. The best results in the respective blocks are underlined and the overall best are bold faced. For the RNN models, the subscript first refer to the input and second the model used. T means that the RNN was trained on text, and LDA / LSA refers to what $\widehat{surprisal}$ was part of the input. The numbers refer to the size of the GRU and the number of GRUs. For LDA / LSA / $TextRank$ the number referes either to the highest ranked words used, or the percentage of words used.

In direct comparison, LSA slightly outperforms LDA in precision, recall and F1-values. In contrast, when surprisal is used as part of the input in a RNN, there is no clear winner. The RNNs trained on surprisal alone outperform the baseline models in several cases and four out of five LDA outperforms LSA. Because of the LSA's greater computational economy and 'niceness' compared to LDA, our conclusion is that LSA is a suitable building block of TCM in the service of NLP's keyword extraction application.