NLP Practical: Part II

Simone Teufel[1]

Michaelmas 2019/20

[1]This part of practical based on a design by Helen Yannadoukakis

- Practical Session Today:
  - How to develop the baseline system(s)
  - How to write the (baseline) Report 1 (20%; ticked)
- Practical Session Oct 30:
  - Check performance of your baseline system
  - Move to doc2vec system
  - Some diagnostics
  - Write Report 2 (40%; assessed)
- Nov 8: Submit Report 1 by 4pm
- Practical Session Nov 13: Text understanding
- Nov 22: Receive Feedback on Report 1
- Nov 29: Submit Report 2 by 4pm
- Dec 6: Submit Report 3 by 4pm

What you should have by now

- NB classifier
- code for feature treatment
- SVM Light
- crossvalidation code
- stemming
- Sign test

Validation corpus

- Use of Validation corpus is another guard against overfitting
- Use it for tuning model parameters
  - eg feature frequency cutoff (for SVM BOW)
  - eg setting doc2vec-related parameters (for SVM with neural embeddings)
- Rules: never train nor test on validation corpus
- Here: designate 10% (first fold)

How to use the validation corpus (here)

- Declare fold 1 (n=10 Round Robin Xval) as validation corpus
- You can now set all your parameters to your heart's content on this validation corpus, without risking overtraining.
    - Train on all remaining 90%
    - Test each parameter on the validation corpus
- After parameter setting, run an entirely new experiment, using only the information of what parameters work best.
- This entirely new experiment is a cross-validation as you did before.
- Note: you have lost some data, and your folds are now a bit smaller.

Standard way to use the validation corpus

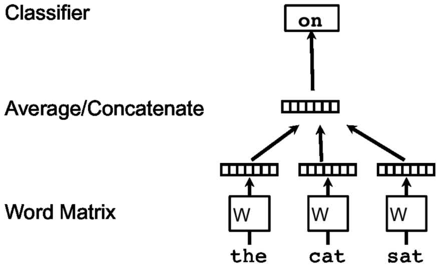
- Work with a 10-10-80 split (validation, test, training)
- Set your parameters by training on the 80% training split
- Choose the best parameters by comparing results on the validation split
- Then test the best system, with the supposedly best parameters, only once, on the test data.
- Not done here, as we want to compare to published cross-validated results.

Doc2vec for Sentiment Analysis

- word2vec: learning neural word embeddings (Mikolov et al., 2013)
- word2vec is a distributional model with dimensionality reduction created on-the-fly, via prediction.
- doc2vec (Le and Mikolov, 2014):[2] embeddings for *sequences* of words
- Agnostic to granularity: sentence, paragraph, document
- Learned 'document' vector effective for various/some tasks, including sentiment analysis

[2] Or paragraph vectors, or document vectors . . .

Task: predict the next word given the context



Optimisation objective:

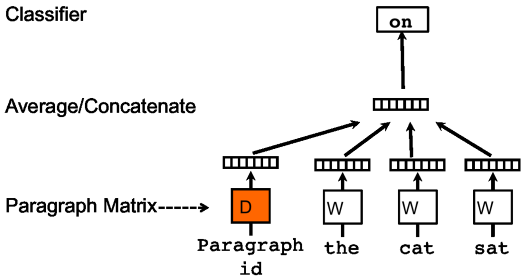
$$\frac{1}{T}\sum_{t=k}^{T-k}\log p(w_t|w_{t-k},\ldots,w_{t+k})$$

Softmax output layer:

$$p(w_t|w_{t-k},\ldots,w_{t+k}) = \frac{\exp y_{w_t}}{\sum_i \exp y_i}$$

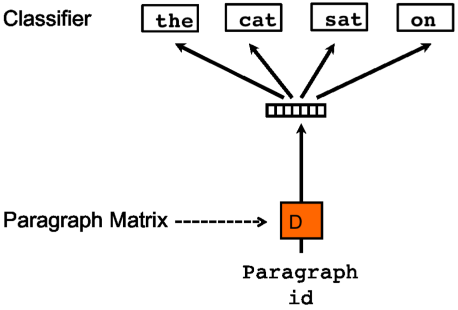
$$y = b + U\,h(w_{t-k},\ldots,w_{t+k};W)$$

Doc2vec: distributed memory (dm) architecture



- Add paragraph token: each paragraph mapped to a unique vector
- Paragraph vector now also contributes to the prediction task
  - Shared across all contexts from the same paragraph
- Works as a “memory” of context / topic

Doc2vec: distributed bag of words (dbow) architecture



Train paragraph vector to predict context words in a window (no word order, given a document vector).
This is similar to word2vec Skip-gram model, which was trained to predict context words given a word vector.

Doc2vec

- Our level of granularity: document / review
- Parameters:
  - Training algorithm (dm, dbow)
  - The size of the feature vectors (e.g., 100 dimensions good enough for us)
  - Number of iterations/epochs (e.g., 10 or 20)
  - Context window
  - Hierarchical softmax (faster version) ...
- Use `gensim` python library

Doc2vec: how can we use it for sentiment analysis?

- Train vectors using a larger 100,000 review corpus (details in instructions)
- Vectors can then be used as features within a typical supervised machine learning framework

A more powerful test: Permutation test

- Paired samples: two systems are run on identical data
- Tests whether the population mean is different (H_1) or the same (H_0)
- Non-parametric tests: no assumptions about distribution in your underlying data
- $$\alpha = P(\text{Type I Error}) = P(\text{Reject}H_0|H_0\text{is True})$$
- α is the probability of a false positive (significance level).
- $$\beta = P(\text{Type II Error}) = P(\text{Do Not Reject } H_0|H_1 \text{ is True})$$
- β is the probability of a false negative. 1-β is the power of the test.

Assumption of Permutation test

- Consider the n paired results of System A and B.
- You will observe a difference d between the means of system A and B.
- If there is no real difference between the systems (and they come from one and the same distribution), it should not matter how many times I **swap** the two results, right?
- There are 2^n permutations (each row can be 0 or 1; swapped or not).
- How many of these permutations result in a difference d as high as the unpermuted version, or higher?
- That proportion is your p
- Final twist: If you cannot test 2^n resamplings, test a large enough random subset

More formally

- The Permutation test evaluates the probability that the observed difference in mean M between the runs has been obtained by random chance.
- If the two runs are indeed the same, then the paired re-assignments should have no impact on the difference in M between the samples.
- Re-sampling: For each paired observation in the original runs, a_i and b_i, a coin is flipped. If 1, then swap the score for b_i with a_i. Otherwise, leave the pair unchanged.
- Repeat R times; compare differences in M.

Monte Carlo Permutation Test

- The probability of observing the difference between the original runs by chance approximated by:

$$p = \frac{s+1}{R+1} \tag{1}$$

  s: number of permuted samples with difference in M higher than the one observed in the original runs
- If $R < 2^n$ because of size, we call this a Monte Carlo Permutation test.

Permutation test: Example with real-valued results

	Original		One permutation		
	System A	System B	Coin Toss	Permuted A	Permuted B
Item 1	0.01	0.1	1	0.1	0.01
Item 2	0.03	0.15	0	0.03	0.15
Item 3	0.05	0.2	0	0.05	0.2
Item 4	0.01	0.08	1	0.08	0.01
Item 5	0.04	0.3	0	0.04	0.3
Item 6	0.02	0.4	1	0.4	0.02
Observed MAP	0.0267	0.205		0.117	0.105
Absolute Observed Difference	0.178		0.0017		

- 2^6 possible permutations for coin throws over 6 items
- Exhaustive resampling: 2 out of 64 permutations are equal or larger than the observed difference in MAP, 0.178.
- p-value$= \frac{2}{64} = 0.0462$.
- Reject Null hypothesis at confidence level $\alpha = 0.05$.

What you should do

- Implement Monte Carlo Permutation test
- Use it in the future for all stat. testing where possible
- Use R=5000

Goal of this Practical – "good science"

- Getting high numerical results isn't everything – neither in this practical nor in science in general
- Good science means:
    - An interesting research question
    - Sound methodology
    - Insightful analysis (something non-obvious)

Goal of this Practical – "good science"

- Getting high numerical results isn't everything – neither in this practical nor in science in general
- Good science means:
  - An interesting research question
  - Sound methodology
  - **Insightful analysis (something non-obvious)**

Insightful analysis

- Finding out what the model is really doing (visualisation via t-SNE, selected / targeted experimentation . . .)
- E.g., see Lau and Baldwin (2016), and Li et al. (2015):
  - Are meaningfully similar documents close to each other?
  - Are document embeddings close in space to their most critical content words?
  - Error analysis – on which documents does SVM misclassify in the worst way? Patterns?

Visualisation example using t-SNE

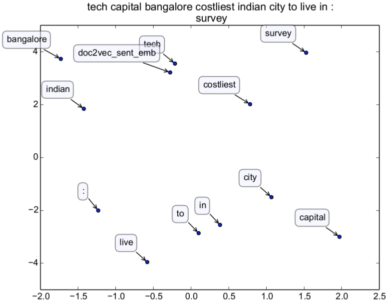


Figure from arxiv.org/abs/1607.05368

From Lau and Baldwin (2016)

Writing tips

- Introduction: pretend this is not a class assignment but your own idea
- Reader has no pre-knowledge
- Describe your data/datasets
- Describe your methodology appropriately
  - Not too detailed (otherwise you look like a beginner)
  - Enough detail for somebody expert (reimplementation)
  - Technical terms: use them – define them first
- Describe your numerical results (after your methods, clearly separated)
- Analyse your numerical results: what is a source of errors? Interpretability of doc2vec space?

Questions?