Introduction to Text Mining

Part X: Practical Issues

Henning Wachsmuth

https://cs.upb.de/css

Practical Issues: Learning Objectives

Concepts

- Text analysis processes and pipelines
- Available libraries and frameworks
- Issues related to effectiveness, efficiency, and robustness

Text analysis techniques

- Joint inference and pipeline extensions
- General effectiveness tweaks
- Pipeline efficiency optimization
- Parallelization
- Domain adaptation and domain independence

Outline of the Course

I. Overview
II. Basics of Linguistics
III. Text Mining using Rules
IV. Basics of Empirical Methods
V. Text Mining using Grammars
VI. Basics of Machine Learning
VII. Text Mining using Similarities and Clustering
VIII. Text Mining using Classification and Regression
IX. Text Mining using Sequence Labeling
X. Practical Issues
   - Text Mining in Practice
   - Effectiveness Issues
   - Efficiency Issues
   - Robustness Issues

Text Mining in Practice

Text Mining in Practice

The real world

- How to develop a text mining approach for a real application?
- How to build up a text mining application?
- What issues to take care of?

From single analyses to analysis processes

- Usually, pipelines of algorithms realize complex analysis processes.
- Many text analysis algorithms are available already.
- Frameworks exist to control the processes.

Main issues in text mining

- Low effectiveness, due to data or approach limitations.
- Low efficiency, due to high run-time or memory consumption.
- Low robustness, due to domain-specific development.

Text Mining in Practice

Development of a Text Mining Approach (Recap)

Input (typical)

- Task. A text mining task to be approached.
- Text corpus. A corpus, split into training, validation, and test set.

A typical development process

1. Analyze on training set how to best tackle the task.
2. Develop (and possibly train) approach that tackles the task.
3. Evaluate the performance of the approach on the validation set.
4. Repeat steps 1–3 until performance cannot be improved anymore.
5. Evaluate the performance of the final approach on the test set.

Output

- Approach. A text mining approach to tackle the given task.
- Results. Empirical performance measurements of the approach.

© Wachsmuth 2020

Text Analysis Processes

What is a text analysis (recap)?

- A text analysis usually infers one specific type of information from text.
  Some also infer a few related types at the same time.

  Tokenization infers tokens   sentiment analysis infers a polarity or score   ...

- Technically, a text analysis adds annotations to a text.

Why text analysis *processes?*

- Many analyses require as input the output of other analyses, which in turn depend on further analyses, and so forth.

  Sentiment analysis might need part-of-speech tagging, which requires tokenization, ...

- Even a single information type may require several analysis steps.
- Most real-world text mining tasks aim at combinations of different types, such as those from information extraction.
- Due to the interdepencies, the standard approach to realize an analysis process is in form of a *text analysis pipeline*.

Text Analysis Processes

Example: Information Extraction (Recap)

What is information extraction?

- Information extraction aims to find entities, relations between entities, and events the entities participate in.
- Input. Unstructured natural language texts.
- Output. Structured information that can, e.g., be stored in databases.

Example task: Extraction of companies' founding dates

Time entity **Organization entity**

" 2014 ad revenues of Google are going to reach

Reference **Time entity**

$20B . The search company was founded in '98 .

Founded relation

Reference **Time entity**

Its IPO followed in 2004 . [...] "

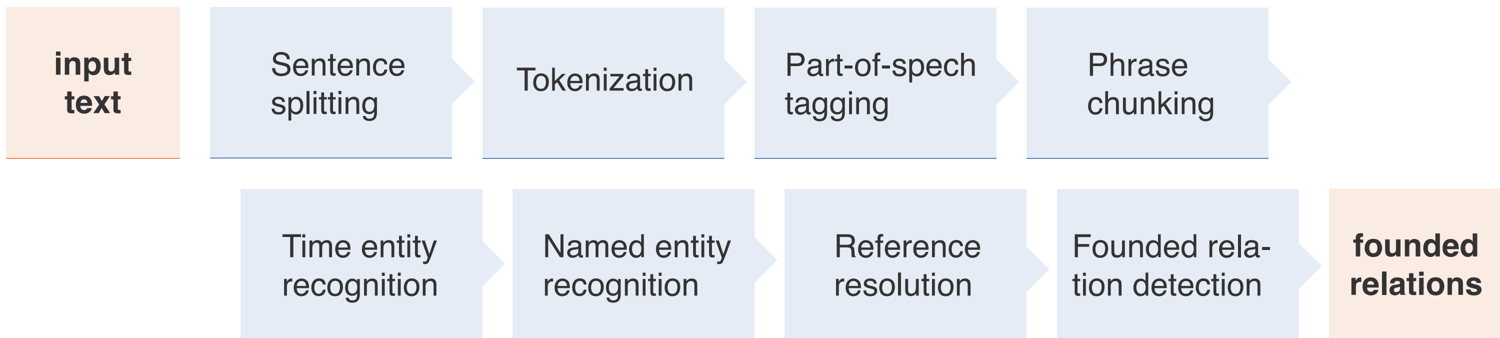
Output: Founded("Google", 1998)

Text Analysis Processes
Text Analysis Pipelines

What is a text analysis pipeline?

- A set of algorithms along with a schedule that defines the order of algorithm application.
- Each algorithm takes as input a text and the output of all preceding algorithms, and it produces further output.

Example pipeline for companies' founding dates



Pipeline scheduling

- The input requirements of each algorithm need to be fulfilled.
- Some analyses are independent, i.e., they have no defined ordering.

Text Analysis Processes

Algorithm Libraries

Problem?

- Tens of algorithms may be needed in a text mining application.
- Implementing all of them from scratch would take forever.

Solution: Algorithm libraries

- Usually, only (or mainly) those algorithms are developed newly that infer the desired output information types in a given task.
- Other algorithms are taken are from available algorithm libraries.
  This includes one's own algorithms from previous text mining applications.
- The decomposition of a process into several analysis steps is a main advantage of the pipeline approach in this regard.

Selected libraries

- Java. Stanford CoreNLP, OpenNLP, LingPipe, mate-tools, TT4j
- Python. Stanford CoreNLP, NLTK, spaCy, Gensim, polyglot

Text Analysis Processes

Text Analysis Frameworks

Problem?

- The data and control flow may be complex in a text mining application.
- Implementing it from scratch is error-prone and time-intensive.

Solution: Text analysis frameworks

- Frameworks that define a standardized way of doing text analysis.
- Algorithms need to match a specific interface.
- Data and control flow handled automatically (very few lines of code).
- The most known frameworks are Apache UIMA and GATE (both Java).

Apache UIMA

- Each algorithm implements a process method.
- Descriptor files specify input and output annotation types of algorithms.
- A pipeline is simply defined as a list of descriptor files.
- The framework calls the process method of each algorithm in a pipeline.

Effectiveness Issues

Effectiveness Issues

General reasons for limited effectiveness

- Ambiguity of natural language.

  "Death penalty — why not?" $\rightarrow$ Stance on death penalty?

- Missing context and world knowledge.

  "I hope Trump will rethink his attitude towards capital punishment." $\rightarrow$ And here?

Process-related reasons for limited effectiveness

- Accumulation of errors through the text analysis process.
- Lack of sufficient training data.
- Domain trainsfer of an approach (see robustness issues further below).

Perfect effectiveness?

- Noisy texts, errors in the ground-truth, subjective tasks, etc. prevent this.

  "i have mixed feelings about the death penalty." $\rightarrow$ Negative stance?

- Only trivial tasks can generally be solved perfectly.

  "Capital punishment KILLS INNOCENT people." $\rightarrow$ Capitalized tokens?

Effectiveness Issues

Accumulation of Errors

What is error accumulation?

- Errors propagate through a text analysis pipeline, since the output of one algorithm serves as input to subsequent ones.
- In standard pipelines, algorithms cannot fix errors of predecessors.
- Even when each algorithm works well, overall effectiveness may be low.

Example from the course

- Automatically classified vs. ground-truth local sentiment.

  Subjectivity accuracy 78.1%, polarity accuracy 80.4%
- Root mean squared errors (RSME) of global sentiment scoring:

Feature type	Automatic	Ground-truth
Local sentiment distribution	0.99	0.77
Discourse relation distribution	1.01	0.84
Sentiment flow patterns	1.07	0.86
Content and style features	1.11	1.11
Combination of features	**0.93**	**0.75**

Effectiveness Issues

Approaches to Counter Error Accumulation

Joint inference algorithms

- Infer multiple information types simultaneously, in order to find the optimal solution over all types.

  In deep learning contexts, a related idea is called *multi-task learning.*
- Knowledge from each task can be exploited for the others.

  Named entity recognition. Avoid confusion between different entity types.

  Argument mining. Segment and classify argument units in one step.
- Reduces run-time efficiency notably and limits reusability.

Pipeline extensions

- Iterative pipelines. Repeat pipeline execution and use the output of later algorithms to improve the output of earlier ones.
- Probabilistic pipelines. Optimize a probability model based on different possible outputs and/or confidence values of each algorithm.
- Both require modifications of algorithms and notably reduce efficiency.

Effectiveness Issues

Lack of Sufficient Training Data

No training data available?

- Hand-crafted rules are the only way to go.
- Careful human tuning on some validation set is important.

Only a small amount of training data?

- If any, use of a “high-bias” learning algorithm, such as Naïve Bayes.
- Also, semi-supervised learning methods may help.

  Or: Find “clever” ways to get more data, such as crowdsourcing or serious games.

A reasonable amount of training data?

- Suitable for advanced feature-based learning algorithms, e.g., SVMs.
- If interpretability is needed, decision trees should be preferred.

A huge amount of training data?

- SVMs and particularly neutral networks may achieve high effectiveness.
- But algorithms such as Naïve Bayes scale much better.

  With enough data, the learning algorithm often matters less.

Effectiveness Issues

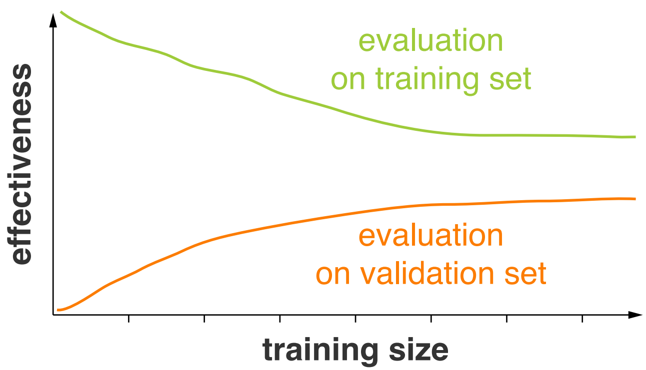
Needed Amount of Data

How much training data is needed?

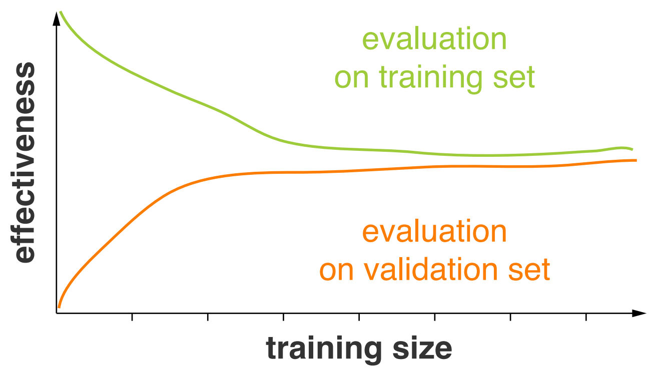
- In general, hard to say.
- Depends on the task, the heterogeneity of the data, ...

One way to find out

- Test different training sizes.
- Evaluate effectiveness on training set and on validation set.



(more data is likely to help)



(more data likely won't help)

- Validation effectiveness is unlikely to ever exceed training effectiveness.

Effectiveness Issues

Practical Effectiveness Tweaks

Exploiting domain knowledge

- Rule of thumb. The narrower the domain, the higher the effectiveness.
- Domain-specific features and weights are very important in practice.
- In-domain training is a must for high effectiveness (so far?).

Combining statistics and rules

- Real-world text mining applications mostly combine statistical learning with hand-crafted rules.
- Rules are derived from a manual review of uncertain and difficult cases.

Scaling up

- At large scale, precision can be preferred over recall, assuming that the information sought for appears multiple times.
- A smart use of redundancy increases confidence.

  "In 1998, they founded Google." "Google exists since 1998." "Google, estd. 1998."

Efficiency Issues

Efficiency Issues

Reasons for limited efficiency

- Large amounts of data may need to be processed, possibly repeatedly.
- Complex, space-intensive models may be learned.
- Text mining often includes several time-intensive text analyses.

Ways to improve memory efficiency

- Machine learning models with high bias are usually smaller.
- Scaling up is the natural solution to higher memory needs.

  Memory efficiency is often *not* the main problem.

Ways to improve run-time efficiency

- Indexing of relevant information.
- Resort to simpler text analysis algorithms.
- Filtering and scheduling in pipelines.
- Parallelization of analysis processes.

  Details on all of them below.

Efficiency Issues

Potential Memory Efficiency Issues

Memory consumption in text mining

- Permanent and temporal storage of input texts and output information.
- Storage of algorithms and models during execution.

Storage of inputs and outputs

- Single input texts are usually small in text mining.
- Output information is negligible compared to input.
- The main problem may be the permanent storage of full text corpora.
  Some take only a few MB's, but large-scale corpora have hundreds of GB's or more.

Storage of algorithms and models

- Some machine learning models take hundreds of MB's of space.
- Word embedding models and similar often take GB's.
- Memory consumption may add up in longer text analysis pipelines.
  Powerful machines are needed — or parallelization.

Efficiency Issues
Indexing and Simpler Algorithms

Indexing of relevant information

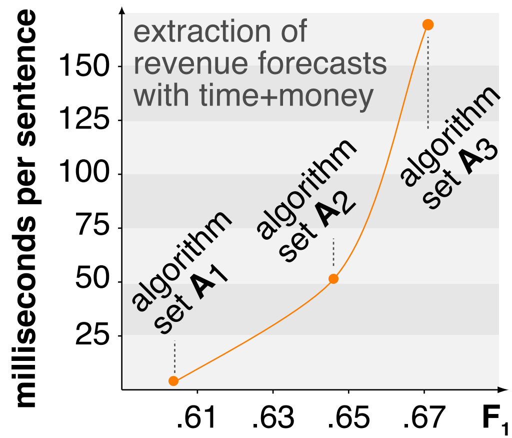
- In applications such as web search, the same information may have to be obtained multiple times from a text.
- By storing and indexing information beforehand, the need for *ad-hoc* text analysis can be avoided.
- Naturally, this is restricted to anticipated information needs.
- Implies a trade-off between run-time and memory efficiency.

Simpler algorithms

- A natural way to improve run-time is to use simpler but faster text analysis algorithms.
- Large efficiency gains possible.

  Recall the k-means results in author clustering.
- At large scale, high effectiveness is possible via redundancy and precision focus.

  See effectiveness issues above.



Efficiency Issues

Filtering in Pipelines

Filtering relevant portions of text

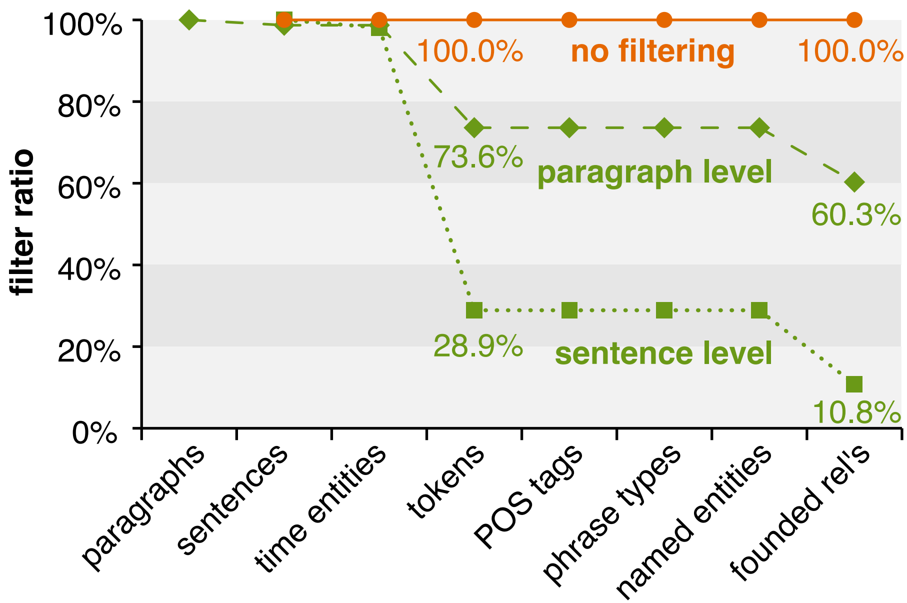
- Standard pipelines apply each algorithm to the whole input.
- For a given text mining task, not all portions of a text are relevant.
- After each analysis step, irrelevant portions can be filtered out.
- The size of the portions trades efficiency for effectiveness.

Example: Extraction of founding dates

- Data. CoNLL'03 test set with 231 news articles.

Some sentence-level results

- Less than 30% tokenized.
- Relation extraction only on every 9th sentence.



Efficiency Issues

Pipeline Scheduling

Optimal scheduling of pipelines

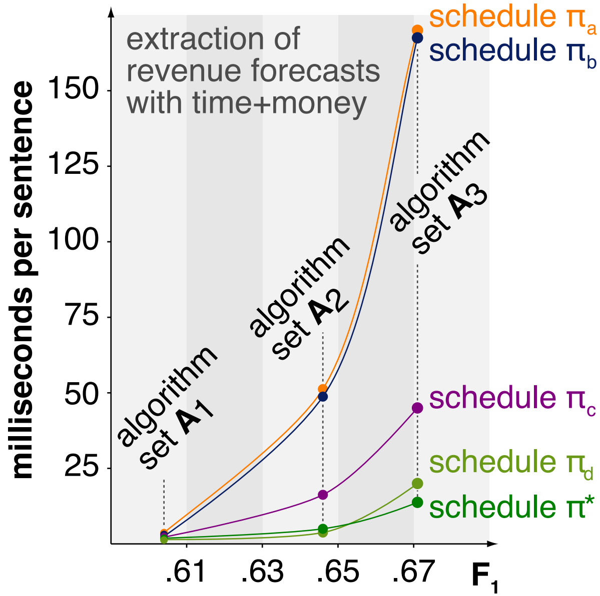
- With filtering, the schedule of a pipeline's algorithms affects efficiency.
- Schedule optimization is a dynamic programming problem based on the run-times and "filter rates" of the algorithms.

Intuition

- Filter out many portions of text early.
- Schedule expensive algorithms late.

Effects

- Efficiency may be improved by more than an order of magnitude.
- If filtering matches the level on which the algorithms operate, effectiveness is maintained.



Efficiency Issues

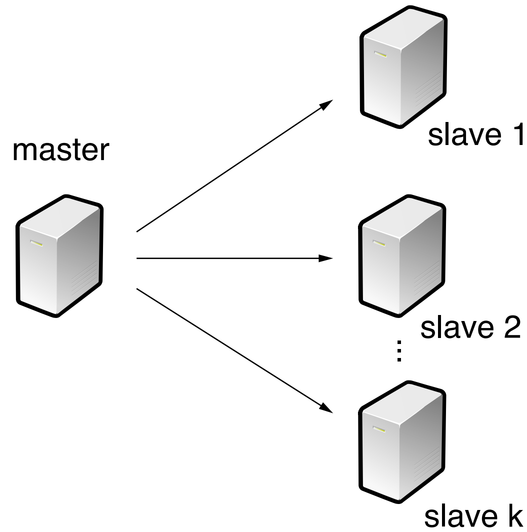
Parallelization

Text analysis entails "natural" parallelization

- Input texts are usually analyzed in isolation, allowing their distribution.

  Focus here on basic scenarios and homogeneous architectures.

Basic parallelization scenario



- One master machine, many slaves.
- Master sends input to slaves.
- Slaves process input and produce output.
- Master aggregates output.

Homogeneous parallel system architecture

- All machines comparable in terms of speed etc.
- No specialized hardware.

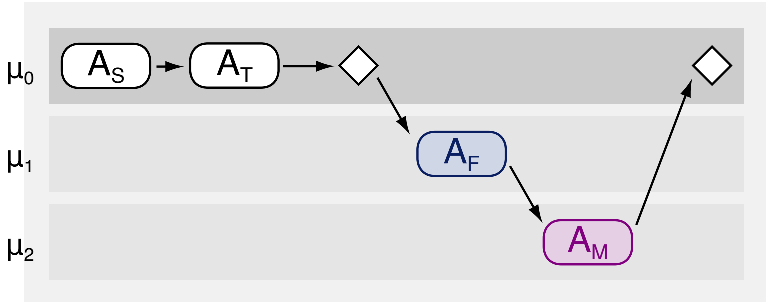
  Heterogenous architectures would require more tailored optimizations.

Efficiency Issues

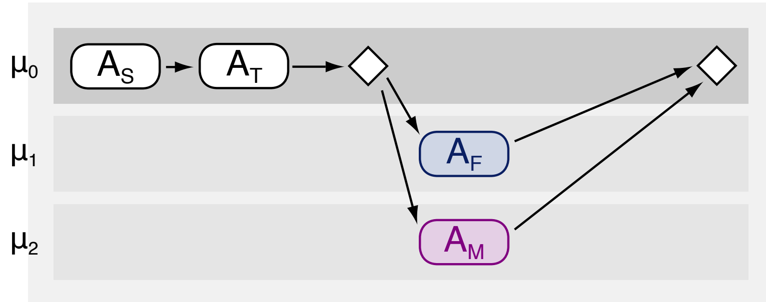
Four Approaches to Parallelizing Text Analysis

parallelization of text analysis algorithms

(a) Analysis pipelining

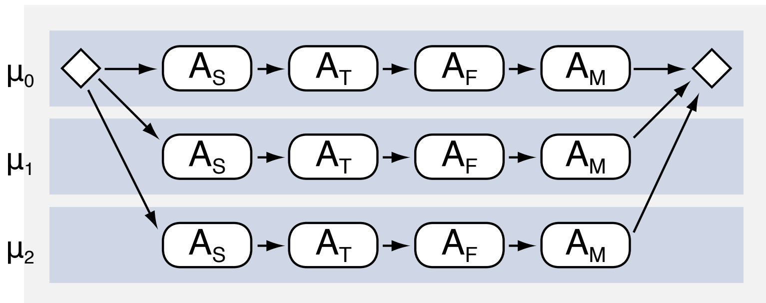


(b) Analysis parallelization

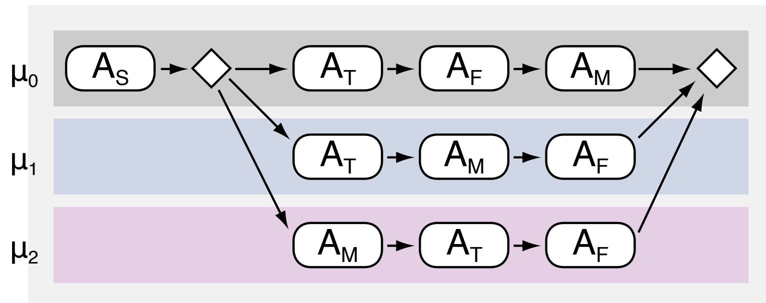


parallelization of text analysis pipelines

(c) Pipeline duplication



(d) Schedule parallelization



(machines μ_0, μ_1, μ_2; text analysis algorithms A_S, A_T, A_F, A_M; A_F and A_M independent)

Efficiency Issues

Discussion of the Parallelization Approaches

Analysis pipelining

- Pro. Low memory consumption, lower execution time.
- Con. Not fault-tolerant, high network overhead, machine idle times.

Analysis parallelization

- Pro. Low memory consumption, possibly lower execution time.
- Con. Not fault-tolerant, network overhead, high machine idle times.

Pipeline duplication

- Pro. Very fault-tolerant, no idle times, much lower execution time.
- Con. Full memory consumption on every slave.

Schedule parallelization

- Pro. Fault-tolerant, few idle times, lower memory consumption, much lower execution time.
- Con. Some network overhead, more complex process control.

Robustness Issues

Robustness Issues

What is (domain) robustness?

- Text mining often needs to be applied to texts with unknown properties.
- Robustness means here that it is effective on texts across *domains*.

Domain

- A set of texts that share certain properties.
- Can refer to a topic, genre, style, ... — or combinations.

  Also, languages are a kind of specific domains, with few feature overlap.
- In general, texts from a domain can be seen as being drawn from the same underlying feature distribution.

  This means that similar feature values imply similar output information.

Reasons for limited domain robustness

- Inclusion of domain-specific features or rules.
- Training on a biased dataset.
- Too much variance in the learned model, i.e., overfitting.

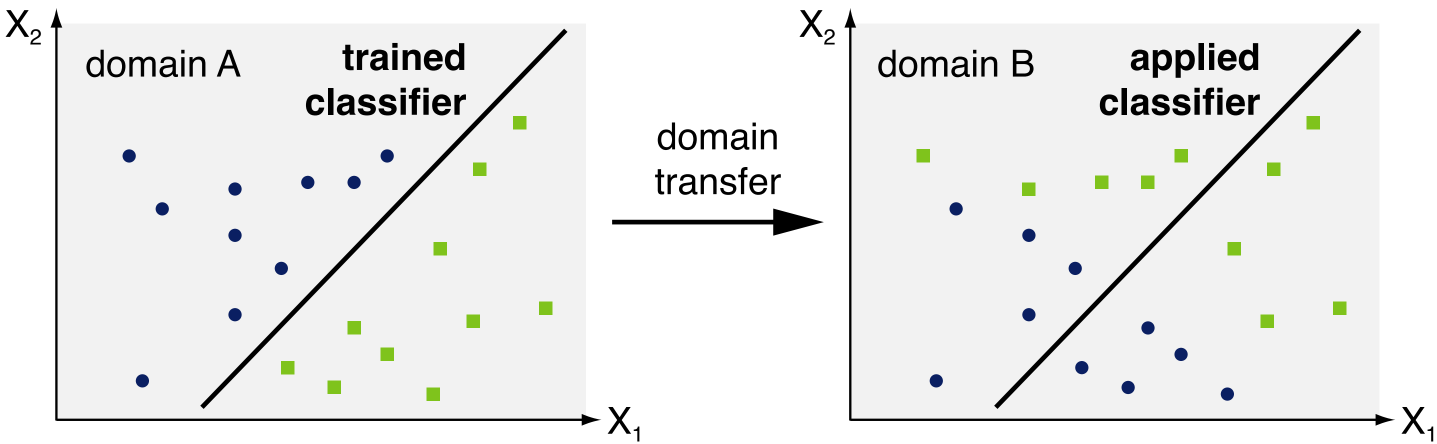
  Missing domain robustness is a fundamental problem of data analysis in general.

Robustness Issues

Domain Dependency

What is domain dependency?

- If a text analysis works (notably) better in the domain of training texts than in others, it is said to be domain-dependent.



Differences between domains

- The same feature values result in different output information.
- Different features are discriminative regarding the target variable.

"Read the book" in book reviews vs. movie reviews... vs. hotel reviews?

Robustness Issues

General Approaches to Improve Domain Robustness

Heterogeneous training sets

- A simple way to make an approach more robust is to train it on texts from multiple (notably different) domains.
- Avoids overfitting to domain-specific features.
- In in-domain settings, typically worse than domain-specific approaches.

Domain-independent features

- For many tasks, there are features that behave similar across domains.

  unambiguous polarity indicators in sentiment analysis,
  spaces in tokenization, ...

- By focusing on such features, robustness can be improved.
- Features that model structure or style (rather than content) tend to be more domain-independent, but exceptions exist.
- The sentiment flow patterns from lecture parts VII+VIII follow this idea.

  See evaluation below.

© Wachsmuth 2020

Robustness Issues

Domain Adaptation

What is domain adaptation?

- The adjustment of an approach developed on some source domain to work well in some target domain.
- Requires at least a few training texts from the target domain.
- Often based on *structural correspondences* of the domains.

Learning of structural correspondences

- Identify features that work robustly across source and target domain.

  "horrible" is likely to be negative in every review.

- Learn correspondence of domain-specific features from cooccurrence with domain-robust features.

  "Read the book!" occurs at the end of movie reviews with "horrible".
  "Stay away!" occurs at the end of hotel reviews with "horrible".

- Align domain-specific features based on correspondences.

  "Read the book!" in movie reviews $\sim$ "Stay away!" in hotel reviews

Review Sentiment Analysis *

Sentiment classification of reviews (recap)

- Classification of the nominal sentiment polarity or score of a customer review on a product, service, or work of art.

Data

- 2100 English hotel reviews from TripAdvisor, scores $\in$ {1, ..., 5}.
  Below, 1–2 mapped to 0, 3 to 1, 4–5 to 2.
- 5,006 English movie reviews from Rotten Tomatoes, scores $\in$ {0, 1, 2}.
  From the Cornell movie review dataset (Pang and Lee, 2005).

Tasks

- 3-class sentiment classification across domains.

Approach

- Algorithm. Linear SVM with one-versus-all multi-class handling.
  Default cost hyperparameter value ($C = 1.0$) in cross-domain evaluation.
- Features. Same as in lecture part VIII.

Review Sentiment Analysis *

Evaluation of Cross-Domain Classification

Evaluation

- SVMs trained on texts from one domain, tested on texts from the other.
- Comparison of in-domain to out-of-domain training on the same test set.

  By comparing on the same test set, the "difficulty" of corpora is ruled out.

Effectiveness results (accuracy)

		Test on Hotel			Test on Movie		
Feature type	**Trained on:**	**Hotel**		**Movie**	**Movie**		**Hotel**
Local sentiment distribution		69.8%	– 11.0	**58.8%**	52.7%	– 12.9	39.8%
Discourse relation distribution		65.3%	– 10.8	54.5%	52.3%	– 7.2	45.1%
Sentiment flow patterns		63.1%	– 6.0	57.1%	53.9%	– 8.6	**45.3%**
Content and style features		58.9%	– 16.1	42.8%	63.8%	– 29.8	34.0%
Combination of features		**71.5%**	– 22.7	48.8%	**64.0%**	– 21.7	42.3%

Observations

- Sentiment flow patterns have smallest effectivess loss across domains.
- Full domain independence seems hard to achieve.

Conclusion

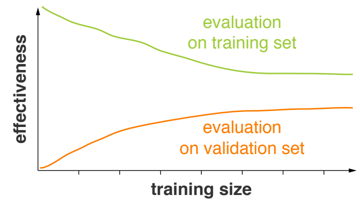
Summary

Practical Issues

- Practical text analysis processes are often complex.
- Algorithm libraries and analysis frameworks help.
- Effectiveness, efficiency, and robustness challenging.

Effectiveness issues

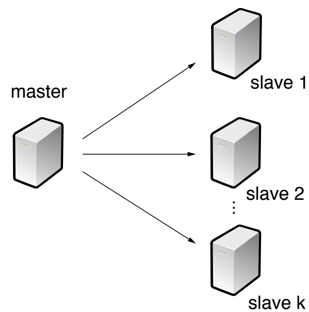
- Error accumulation in the process must be faced.
- Available data size governs algorithm choices.
- Ambiguity and lack of context remain the main issues.



(more data is likely to help)

Efficiency and robustness issues

- Efficiency of analysis processes can be optimized.
- Text mining can be parallelized very well.
- Domain robustness is hard to achieve in general.



References

Some content and examples taken from

- Daniel Jurafsky and Christopher D. Manning (2016). Natural Language Processing. Lecture slides from the Stanford Coursera course. `https://web.stanford.edu/~jurafsky/NLPCourseraSlides.html`.
- Henning Wachsmuth (2015): Text Analysis Pipelines — Towards Ad-hoc Large-scale Text Mining. LNCS 9383, Springer.