



An Introduction to Text Classification

Jörg Steffen, DFKI
steffen@dfki.de

29.10.2012



- Application Areas
- Rule-Based Approaches
- Statistical Approaches
 - Naive Bayes
 - Vector-Based Approaches
 - Rocchio
 - K-nearest Neighbors
 - Support Vector Machine
- Evaluation Measures
- Evaluation Corpora
- N-Gram Based Classification

Example Application Scenario



- Bertelsmann “Der Club” uses text classification to assign incoming emails to a category, e.g.
 - change of bank connection
 - change of address
 - delivery inquiry
 - cancellation of membership
- Emails are forwarded to the responsible editor
- Advantages
 - decrease of response time
 - more flexible resource management
 - happy customers 😊

Other Application Areas !



- Spam filtering
- Language identification
- News topic classification
- Authorship attribution
- Genre classification
- Email surveillance

Rule-based Classification Approaches !



- Use Boolean operators AND, OR and NOT
- Example rule
 - if an email contains “address change” or “new address”, assign it to the category “address changes”
- Organized as decision tree
 - nodes represent rules that route the document to a subtree
 - documents traverse the tree top down
 - leafs represent categories
 - rules are not independent of each other

Rule-based Classification Approaches !



- Advantages
 - transparent
 - easy to understand
 - easy to modify
 - easy to expand
- Disadvantages
 - complex and time consuming
 - intelligence is not in the system but with the system designer
 - not adaptive
 - only absolute assignment, no confidence values
- Statistical classification approaches solve some of these disadvantages



- Use statistics to automatically create decision trees
 - e.g. ID3 or CART
- Idea: identify the feature of the training data with the highest information content
 - most valuable to differentiate between categories
 - establish the top level node of the decision tree
 - recursively applied to the subtrees
- Advanced approaches “tune” the decision tree
 - merging of nodes
 - pruning of branches

Statistical Classification Approaches !



- Advantages
 - work with probabilities
 - allows thresholds
 - adaptive
- Disadvantage
 - require a set of training documents annotated with a category
- Most popular
 - Naive Bayes
 - Rocchio
 - K-nearest neighbor
 - Support Vector Machines (SVM)



- Remove HTML/XML tags and stop words
- Perform word stemming
- Replace all synonyms of a word with a single representative
 - e.g. { car, machine, automobile } → car
- Composites analysis (for German texts)
 - split “Hausboot” into “Haus” and “Boot”
- Set of remaining words is called “feature set”
- Documents are considered as “Bag-of-Words”
- Importance of linguistic preprocessing increases with
 - number of categories
 - lack of training data

Naive Bayes !



- Based on Thomas Bayes theorem from the 18th century
- Idea: Use the training data to estimate the probability of a new, unclassified document $d = \{w_1, \dots, w_M\}$ belonging to each category c_1, \dots, c_K

$$P(c_j | d) = \frac{P(c_j)P(d | c_j)}{P(d)}$$

- This simplifies to
$$P(c_j | d) = P(c_j) \prod_{i=1}^M P(w_i | c_j)$$

Naive Bayes !



- The following estimates can be done using the training documents

$$P(c_j) = \frac{N_j}{N}$$

$$P(w_i | c_j) = \frac{1 + N_{ij}}{M + \sum_{k=1}^M N_{kj}}$$

where

- N is the total number of training documents
- N_j is the number of training documents for category c_j
- N_{ij} is the number of times word w_i occurred within documents of category c_j
- M is the total number of words in the document

Naive Bayes !



- Result is a ranking of categories
- Adaptive
 - probabilities can be updated with each correctly classified document
- Naive Bayes is used very effectively in adaptive spam filters
- But why “naive”?
 - assumption of word independence
 - generally not true for word appearances in documents
- Conclusion
 - Text classification can be done by just counting words

Documents as Vectors !



- Some classification approaches are based on vector models
- Developed by Gerard Salton in the 60s
- Documents have to be presented as vectors
- Example
 - the vector space for two documents consisting of “I walk” and “I drive” consists of three dimension, one for each unique word
 - “I walk” $\rightarrow (1, 1, 0)$
 - “I drive” $\rightarrow (1, 0, 1)$
- Collection of documents is represented by a word-by-document matrix $A = (a_{ik})$ where each entry represents the occurrences of a word i in a document k

Weight of Words in Document Vectors !



- Boolean weighting

$$a_{ik} = \begin{cases} 1 & \text{if } f_{ik} > 0 \\ 0 & \text{otherwise} \end{cases}$$

- Word frequency weighting

$$a_{ik} = f_{ik}$$

- tf.idf weighting

$$a_{ik} = f_{ik} \times \log\left(\frac{N}{n_i}\right)$$

- considers distribution of words over the training corpus
- n_i is the number of training documents that contain at least one occurrence of word i

Dimensionality Reduction !



- Large training corpora contain hundreds of thousands of unique words, even after linguistic preprocessing
- Result is a high dimensional feature space
- Processing is extremely costly in computational terms
- Use **feature selection** to remove non-informative words from documents
 - document frequency thresholding
 - information gain
 - χ^2 -statistic

Document Frequency Thresholding



- Compute document frequency for each word in the training corpus
- Remove words whose document frequency is less than predetermined threshold
- These words are non-informative or not influential for classification performance

Information Gain



- Measure for each word how much its presence or absence in a document contributes to category prediction
- Remove words whose information gain is less than predetermined threshold

$$IG(w) = -\sum_{j=1}^K P(c_j) \log P(c_j) + P(w) \sum_{j=1}^K P(c_j | w) \log P(c_j | w) + P(\bar{w}) \sum_{j=1}^K P(c_j | \bar{w}) \log P(c_j | \bar{w})$$

Information Gain



$$P(c_j) = \frac{N_j}{N}$$

$$P(w) = \frac{N_w}{N}$$

$$P(c_j | w) = \frac{N_{jw}}{N_w}$$

$$P(\bar{w}) = \frac{N_w^-}{N}$$

$$P(c_j | \bar{w}) = \frac{N_{j\bar{w}}}{N_w^-}$$

- N total no. of documents
- N_j no. of docs in category c_j
- N_w no. of docs containing w
- N_w^- no. of docs not containing w
- N_{jw} no. of docs in category c_j containing w
- $N_{j\bar{w}}$ no. of docs in category c_j not containing w



- Measure dependance between words and categories

$$\chi^2(w, c_j) = \frac{N \times (N_{jw}N_{j\bar{w}} - N_{j\bar{w}}N_{jw})^2}{(N_{jw} + N_{j\bar{w}}) \times (N_{\bar{j}w} + N_{\bar{j}\bar{w}}) \times (N_{jw} + N_{\bar{j}w}) \times (N_{j\bar{w}} + N_{\bar{j}\bar{w}})}$$

- Define measure as

$$\chi^2(w) = \sum_{j=1}^K P(c_j) \chi^2(w, c_j)$$

- Result is a word ranking
- Select top section as feature set

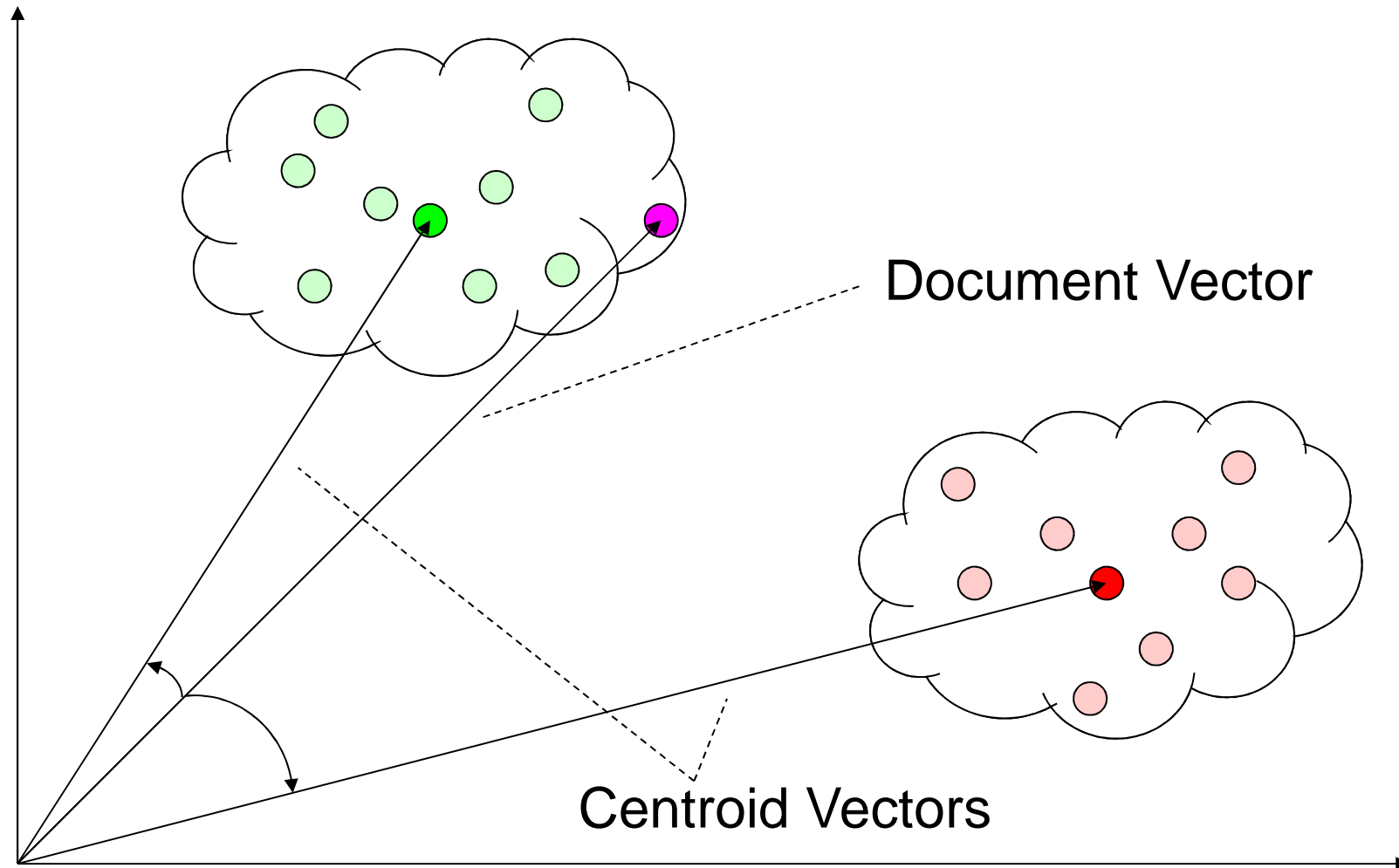
Rocchio !



- Uses **centroid** vectors to represent a category
- Centroid vector is the average vector of all document vectors of a category
- Centroid vectors are calculated in the training phase
- To classify a new document, just calculate its distance to the centroid vector of each category
- Use cosine similarity as distance measure

$$\cos(\vec{x}, \vec{y}) = \frac{\sum_i x_i y_i}{\sqrt{\sum_i x_i^2} \times \sqrt{\sum_i y_i^2}}$$

Rocchio !





- Advantages
 - fast training phase
 - small models
 - fast classification
- Disadvantages
 - precision drops with increasing number of categories

K-nearest Neighbors !

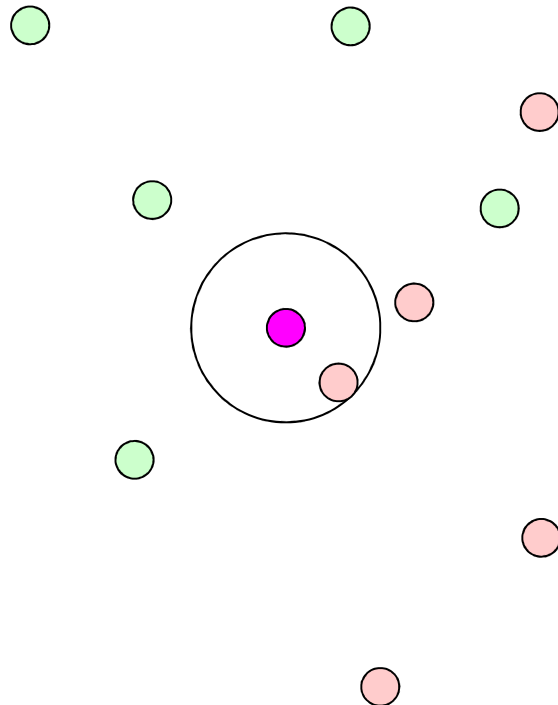


- Similar to Rocchio
- Check the k nearest neighbor vectors of a new document vector
- Value of k determined empirically
- Define “nearest” using a similarity measure, e.g. Euclidean distance or cosine similarity

1-nearest Neighbor !



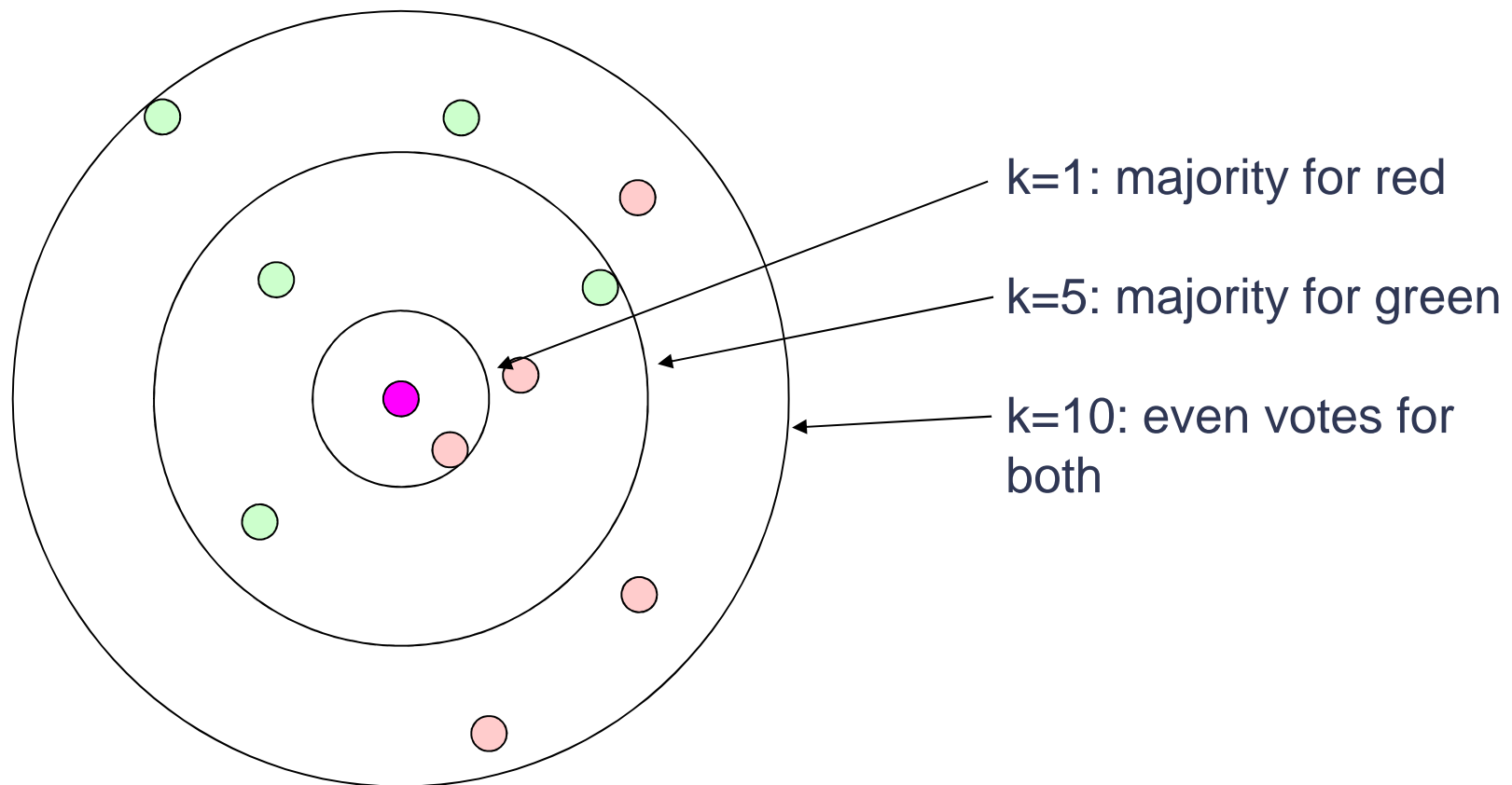
- Assign new document the category of its nearest neighbor



K-nearest Neighbors !



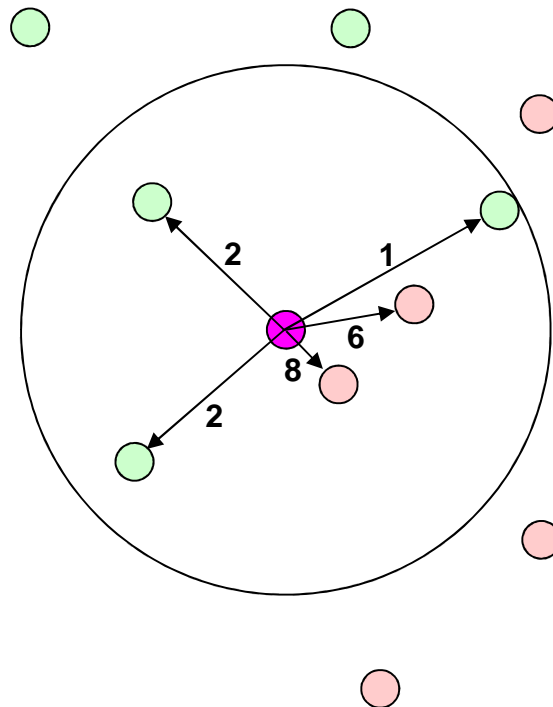
- Majority voting scheme



K-nearest Neighbors !



- Weighted sum voting scheme for $k = 5$
- Neighbors are given weights according to their nearness



weighted sum for red: 14

weighted sum for green: 5

K-nearest Neighbors !



- Advantages
 - no training phase required
 - good scalability if number of categories increases
- Disadvantages
 - large models for large training sets
 - requires a lot of memory
 - slow performance

Support Vector Machine !

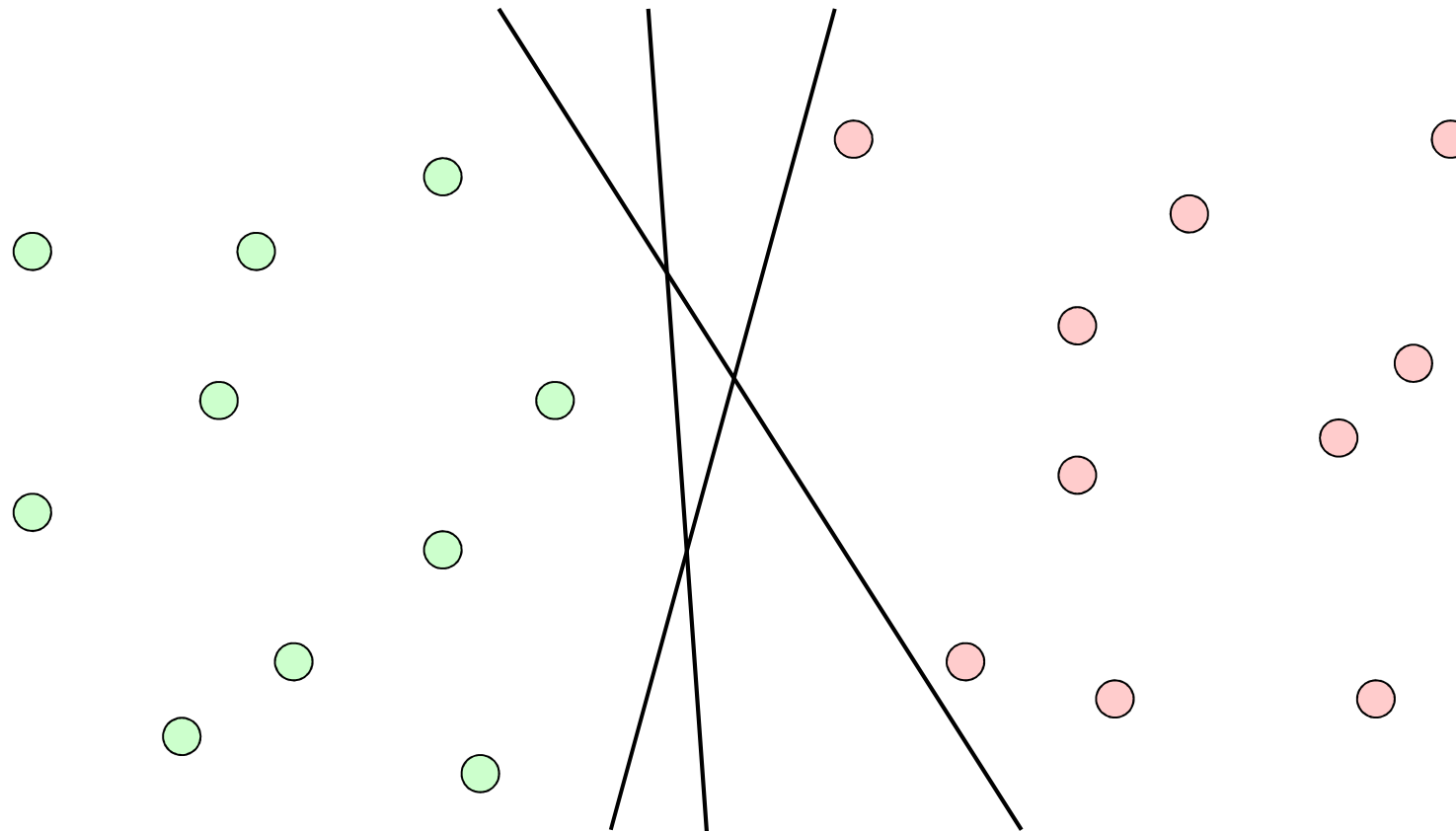


- For each pair of categories find a decision surface (hyperplane) in the vector space that separates the document vectors of the two categories
- Usually, there are many possible separating hyperplanes
- Find the “best” one: **maximum-margin hyperplane**
 - equal distance to both document sets
 - margin between hyperplane and document sets is at maximum
- Training result for each pair of categories: vectors closest to the hyperplane → support vectors
- Classification: calculate distance of document vector to support vectors

Support Vector Machine !



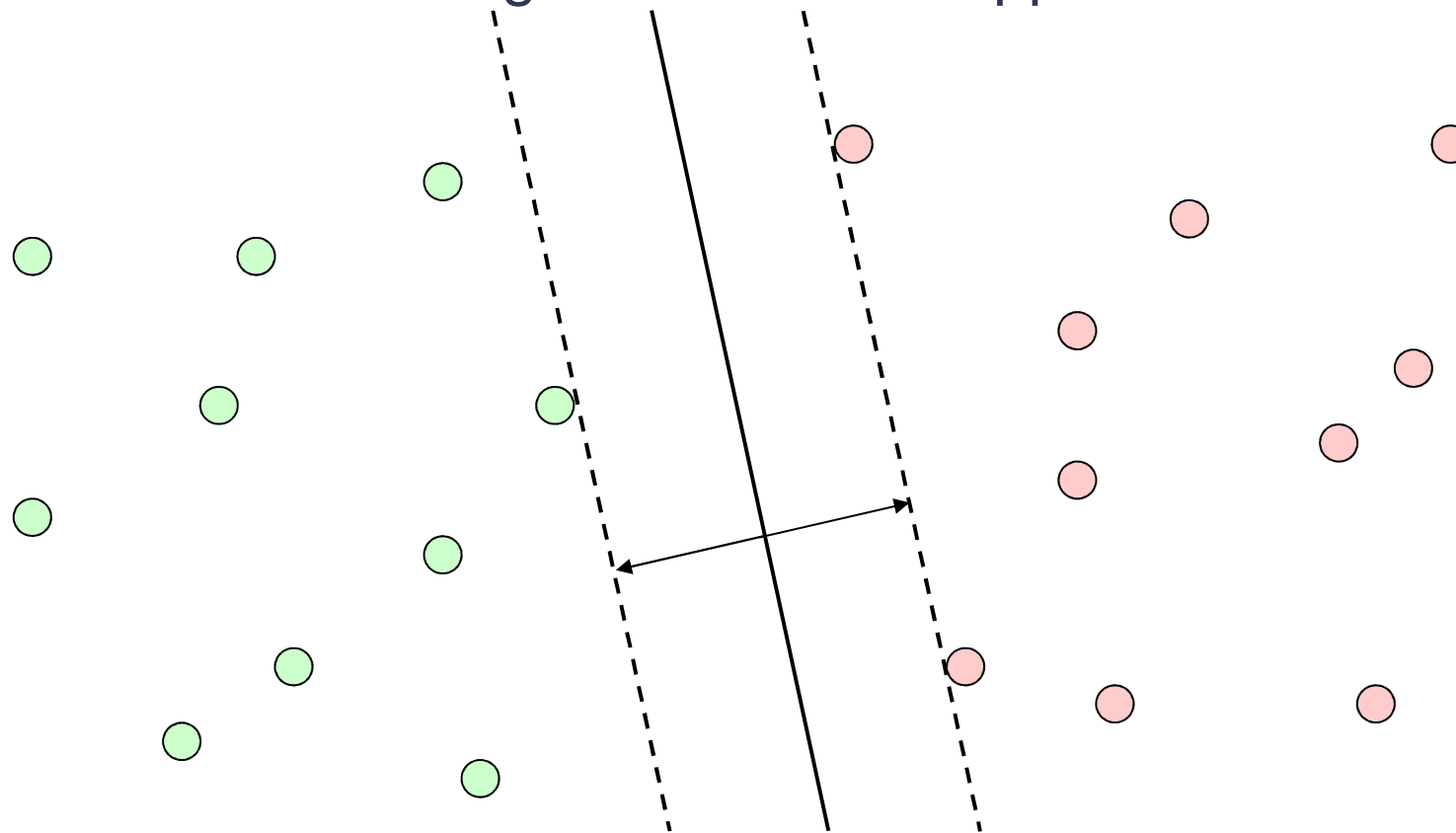
- More than one hyperplane separate the document vectors of each category



Support Vector Machine !



- Find the maximum-margin hyperplane
- Vectors at the margins are called support vectors



Support Vector Machine !



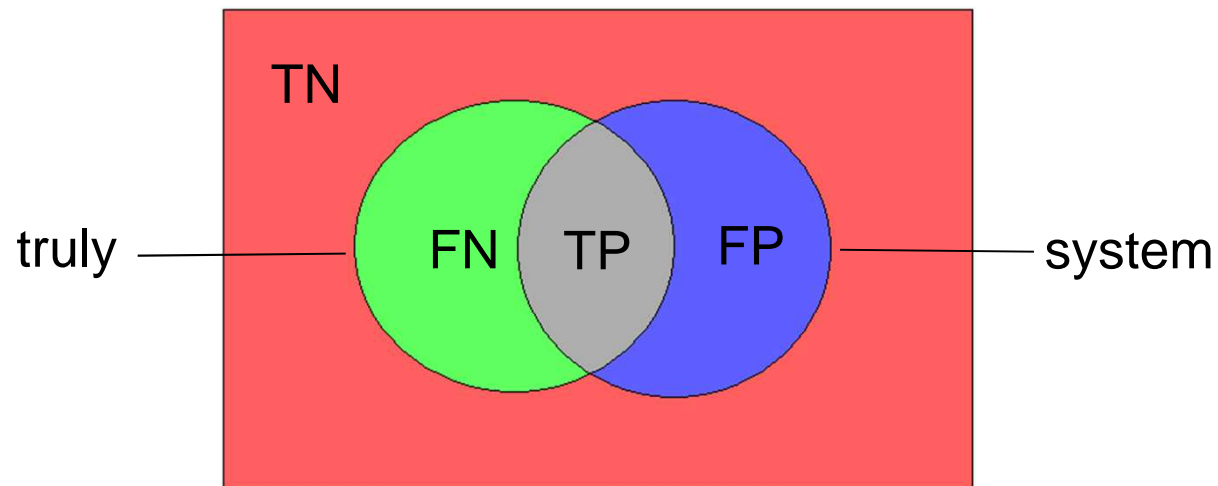
- Advantages
 - only the support vectors are required to classify new documents
 - small models
 - feature selection can be omitted
 - no overfitting
 - when given too much training data, other classification approaches only return a correct classification for training documents
 - main advantage of SVM over other vector-based approaches
- Disadvantage
 - very complex training (optimization problem)

Classification Evaluation !



- Possible results of a binary classification

	truly YES	truly NO
system YES	true positives	false positives
system NO	false negatives	true negatives



Evaluation Measures !



- Precision

- percentage of documents correctly identified as belonging to the category

$$\textit{precision} = \frac{\textit{true positives}}{\textit{true positives} + \textit{false positives}}$$

- Recall

- percentage of documents found belonging to the category

$$\textit{recall} = \frac{\textit{true positives}}{\textit{true positives} + \textit{false negatives}}$$

Evaluation Measures !



- Precision and recall are misleading when examined alone
- There is always a tradeoff between precision and recall
 - Increase in recall often comes with a decrease in precision
 - If precision and recall are tuned to have the same value, it is called the **break-even point**
- F-Measure combines both precision and recall in one value

$$F_{\beta} = \frac{(\beta^2 + 1) \times \textit{precision} \times \textit{recall}}{\beta^2 \times \textit{precision} + \textit{recall}}$$

- β allows different weighting of precision and recall
- for equal weighting, $\beta = 1$



- To compare different classification approaches, a common set of data is required
- Popular evaluation corpora
 - Reuters-21578 collection
 - 20-newsgroup-corpus
- Evaluation corpora are usually split up into a training corpus and a test corpus
- Beware: You can score top precision and recall values if you test your classification approach on the training data!

Reuters-21578 Collection



- Collected from the Reuters newswire in 1987
- Contains 12902 news articles from 135 different categories
- Documents have up to 14 categories assigned
- Average is 1.24 categories per document
- Default split
 - 9603 training documents
 - 3299 test documents

20-Newsgroups-Corpus



- Consists of newsgroup articles from 20 different newsgroups
- Some newsgroups closely related, e.g. alt.atheism and talk.religion.misc
- Contains 20.000 articles, 1000 articles for each newsgroup
- Corpus size: 36 MB
- Average size of article: 2 KB
- Newsgroup header of articles has been removed

What is the best classification approach? !

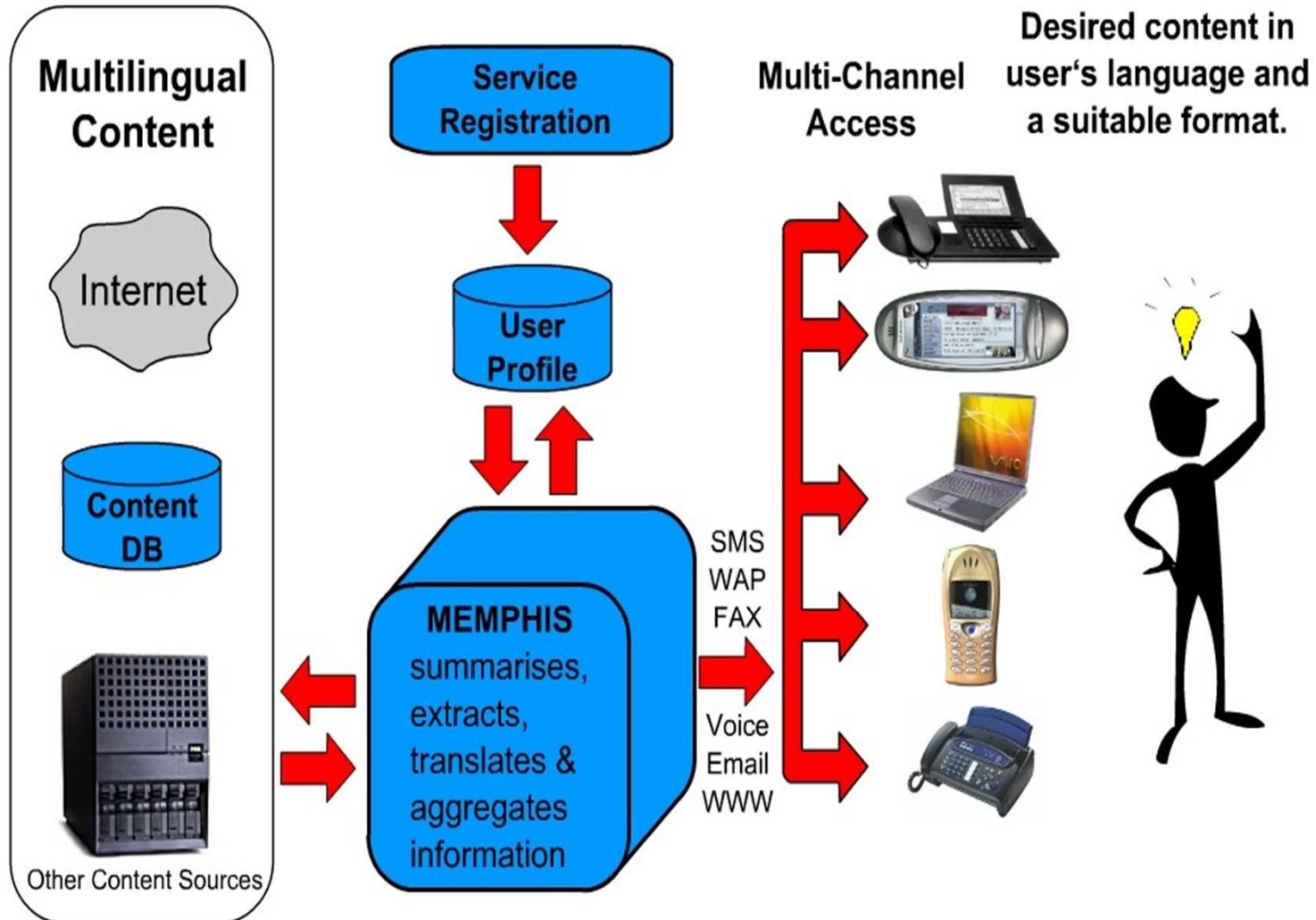


- This depends on the application scenario and the data
- “Hard” facts are easy to model with rules
- “Soft” facts are better modeled with statistic
- If there is few or no training data, statistic doesn’t work
- Among statistical approaches the ranking is
 - SVM
 - K-nearest neighbors
 - Rocchio
 - Naive Bayes
- In real life, rule-based and statistical approaches are often combined to get the best results



N-Gram Based Multilingual and Robust Document Classification

Memphis Project Overview





- Domain: book announcements
- Sources: internet sites of book shops and publishers in English, German and Italian
- Classification task: assign topic to book announcement
 - Biographies
 - Film
 - Music
 - Sports
 - Travel
 - Health
 - Food
- Classification Challenges:
 - Informal texts with open-ended vocabulary
 - Content in several languages
 - Spelling mistakes and missing case distinction



- MEMPHIS classifier based on character-level n-grams instead of terms
- Example
 - “Well, this is an example!”
 - 3-grams: “Wel” “ell” “ll,” “l, ” “, t” “ th” “thi” “his” ... “le!”
- Advantages of character-level n-grams
 - No linguistic preprocessing necessary
 - Language independent
 - Very robust
 - Less sparse data



- Training requires a corpus of documents
- Each training document must be tagged with one or more categories
- For each category, a statistical model is created
- Each model contains conditional probabilities based on character-level n-gram frequencies counted in training documents
- Models are independent of each other



- Document is a character sequence

$$S = c_1, \dots, c_N$$

- Maximum Likelihood Estimate:

$$P(c_i | c_{i-n+1}, \dots, c_{i-1}) = \frac{\#(c_{i-n+1}, \dots, c_i)}{\#(c_{i-n+1}, \dots, c_{i-1})}$$

- Example:

$$P(d | \text{win}) = \frac{\#(\text{wind})}{\#(\text{win})}$$



- Based on Bayesian decision theory
- For each model, predict probability of test document using the chain rule of probability:

$$P(c_1, \dots, c_N) = \prod_{i=1}^N P(c_i | c_1, \dots, c_{i-1})$$

- Approximation in n-gram models:

$$P(c_i | c_1, \dots, c_{i-1}) = P(c_i | c_{i-n+1}, \dots, c_{i-1})$$

- Result is a ranking of categories derived from the probability of the test document in each model



- N-grams in test documents that are unseen in training get zero probability
- As a consequence, probability for test document becomes zero
- No matter how much training data, there can always be unseen n-grams in some test documents
- Solution: Probability Smoothing
 - Assign non-zero probability to unseen n-grams
 - To keep a valid model, reduce the probability of known n-grams and reserve some room in the probability space for unseen n-grams



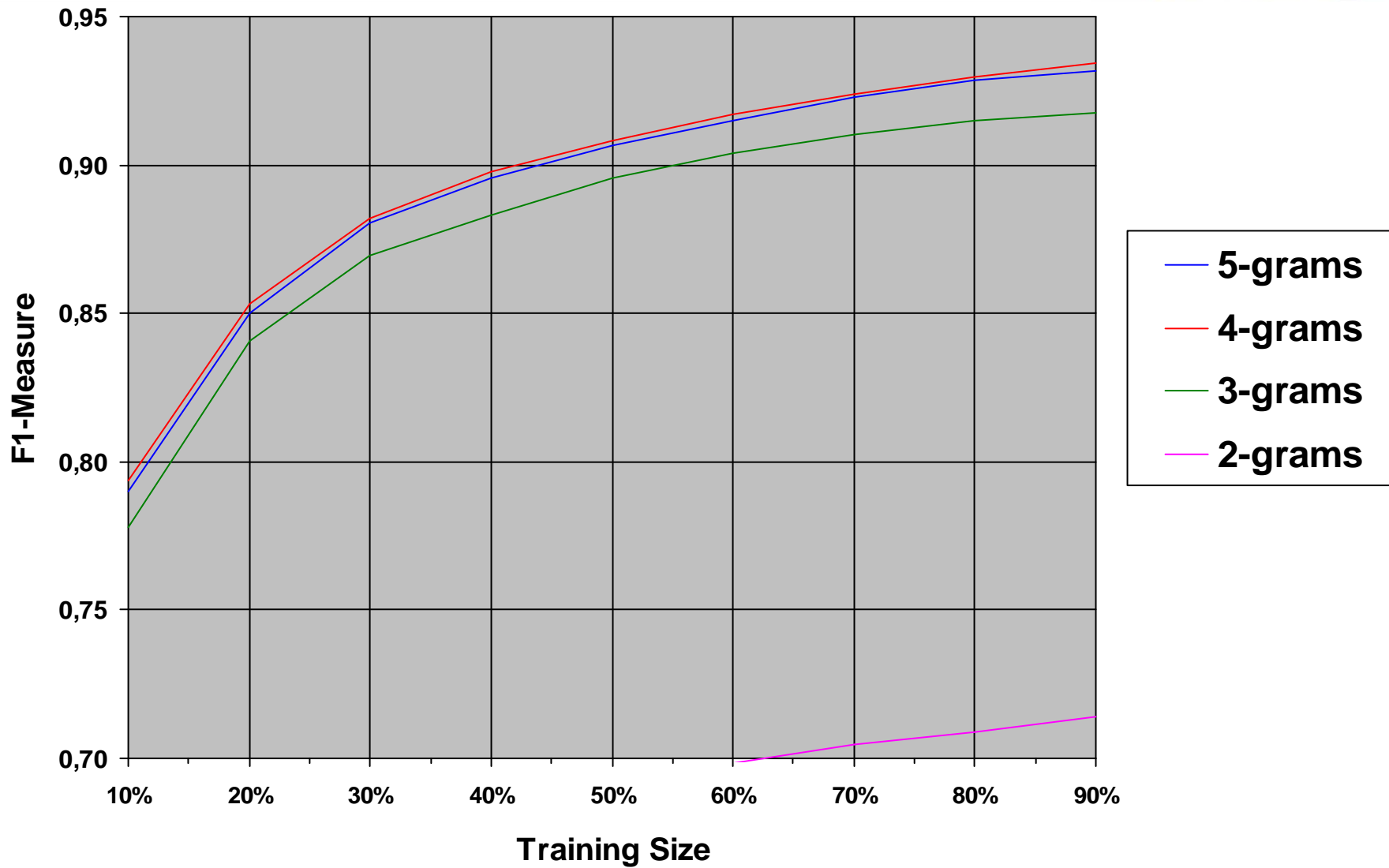
- Several smoothing techniques have been adapted for character-level n-grams that yield backoff models and interpolated models:
 - Katz Smoothing
 - Simple Good-Turing Smoothing
 - Absolute Smoothing
 - Kneser-Ney Smoothing
 - Modified Kneser-Ney Smoothing

Whitespace Stripping



- Non-linguistic preprocessing step
- Strip all whitespaces
- Convert all characters to lower case
- To preserve word border information, first character is always upper case
- Example:
 - LIFE STORIES: Profiles from the New Yorker
 - LifeStories:ProfilesFromTheNewYorker
- Improves average F_1 -Measure by up to 5%
- Larger models

20-Newsgroups Evaluation Results





- Amazon corpora
 - 1000 docs per category
 - English (13MB) and German (10MB)
 - Acquired using the Amazon web service
- Other English corpora:
 - Randomhouse.com (3000 docs, 4 MB)
 - Powells.com (8000 docs, 7MB)
- Other German corpora:
 - Bol.de (1200 docs, 1 MB)
 - Buecher.de (2300 docs, 2 MB)

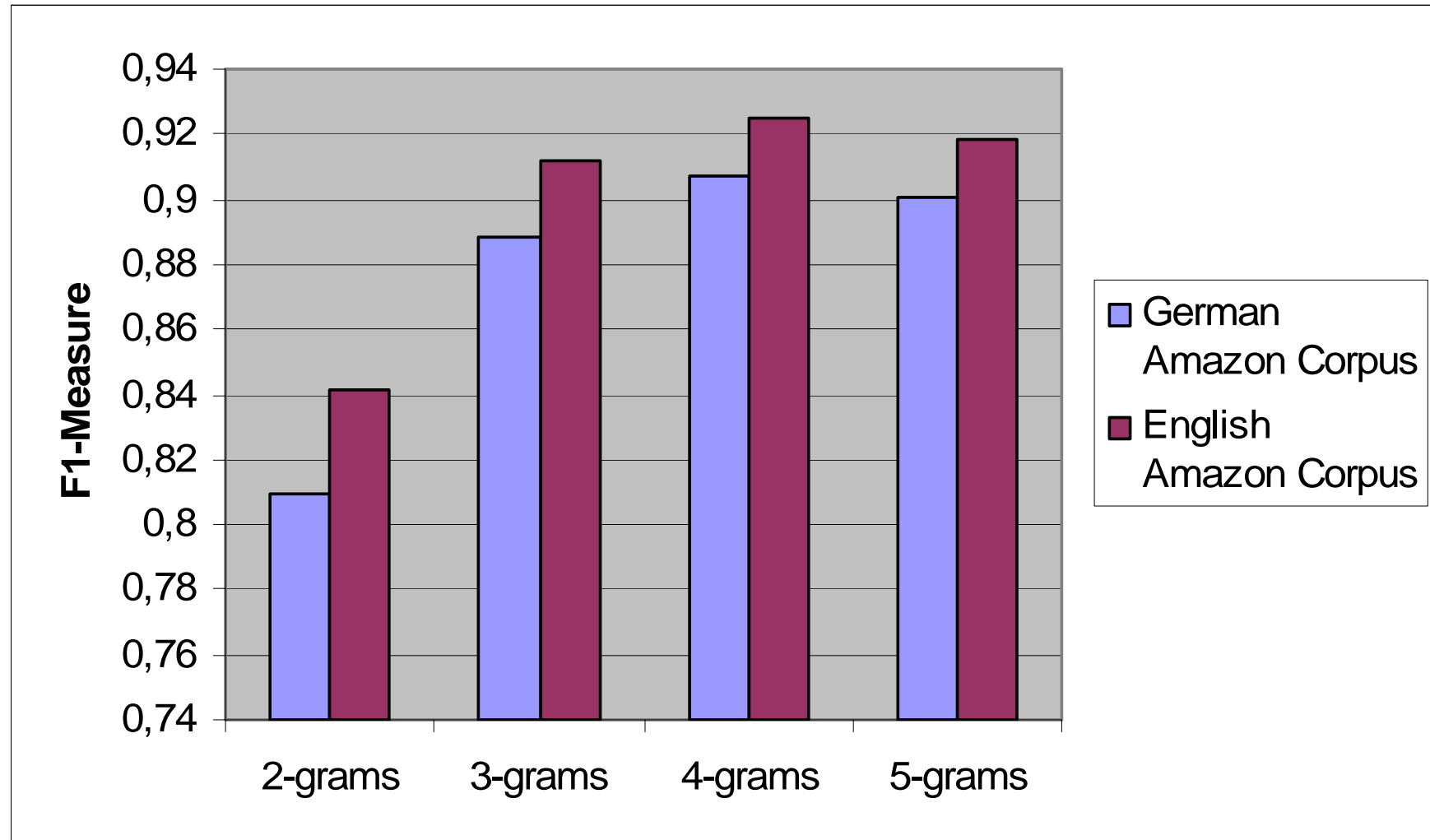


- Classification parameters
 - Smoothing technique
 - N-gram length
 - Mono-lingual vs multi-lingual models
- Setting:
 - Split corpus randomly into training docs (80%) and test docs (20%)
 - Performance as average F_1 -Measure of 10 runs

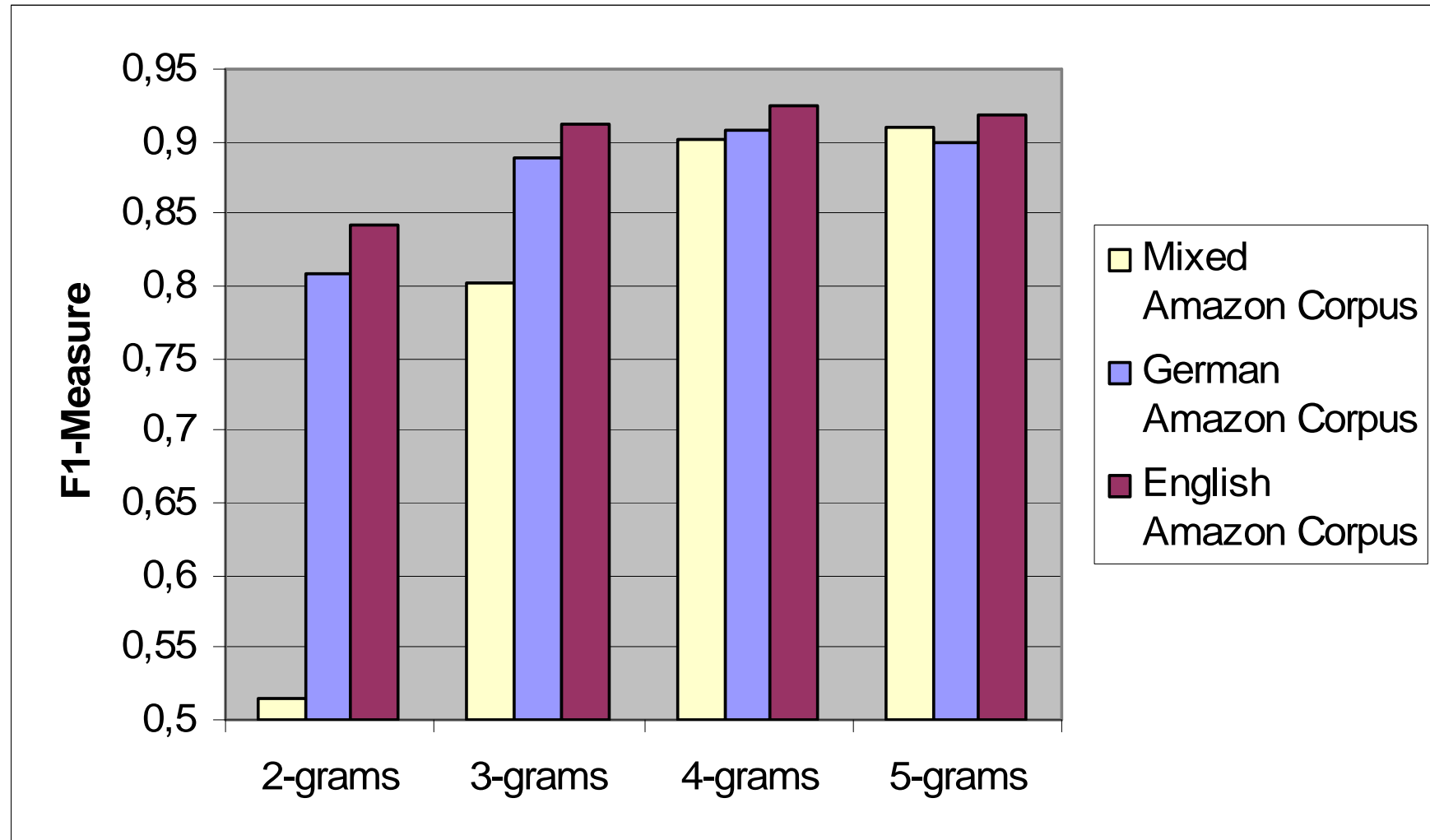
Smoothing Techniques



Mono-Lingual Models



Multi-Lingual Models



Conclusions



- Classification using character-level n-grams performs very good in assigning topics to multi-lingual, informal documents
- Approach is robust enough to allow multi-lingual models