CSC 390
Topics in Artificial Intelligence

"Unsupervised Machine Learning"

Fall 2016

Prof. Sara Mathieson
Smith College

Outline: 11/1

- Today:
  - **Jessica Tin** 10:30am
  - **Jackie** 10:45am
  - **Li** 11:00am
  - **Youyou** 11:15am
  - MSA (mid-semester assessment) 11:30am

- Office Hours today (CSC240): 4-5pm, Ford 346

- Presenters:
  - Speak loudly
  - I will give you a 2 min warning after 10 minutes

- Audience:
  - Give presenters your full attention and ask questions
  - I will ask questions too

“Understand Short Texts by Harvesting and Analyzing Semantic Knowledge”

Wen Hua, Zhongyuan Wang, Haixun Wang, Kai Zheng, and Xiaofang Zhou, 2016

Jessica Tin
CSC 390
November 1, 2016

Motivation

- Short texts are increasingly prevalent — microblogging (Twitter), web search

- Short texts do not have the same properties as longer forms
  - Irregular syntax
  - Less content
  - Ambiguity and noise
  - Enormous volume

  ➛ *traditional Natural Language Processing methods are insufficient*

- Need to use **semantic knowledge**

Key Terminology

- **Short texts** — texts with limited content (e.g. tweets, search queries)

- **Short text understanding** — detecting concepts mentioned in a short text
  1. Text segmentation — divide a text into a sequence of terms
  2. Type detection — determine term types (e.g. POS), recognize instances
  3. Concept labeling — infer the concept of each instance

- **Semantic knowledge** — information concerning meaning

	Definition	**Example**
s	short text	book hotel california
p	segmentation	{book hotel california}
t	term	hotel,california,hotel california
$\bar{t}$	typed-term	$\text{book}_{[v]}$,$\text{book}_{[c]}$,$\text{book}_{[e]}$
$\bar{t}.r$	type	v,adj,att,c,e
$\bar{t}.\vec{C}$	concept cluster vector	({theme park,park},{company}...)

Databases/Corpora

- **YourDictionary**: list of English verbs, adjectives

- **Probase**: semantic network of concepts (e.g. *country*), instances (e.g. *china*), and attributes (e.g. *population*), over 20.7 million instance–concept mappings
  - isA and isAttributeOf relationships
  - Built from corpus of 1.68 billion webpages
  - Quantifies Popularity $p(c|e)$, Typicality $p(e|c)$

- **Wikipedia**: list of synonyms (from redirect links, disambiguation links, hyperlnks between articles)

Methods

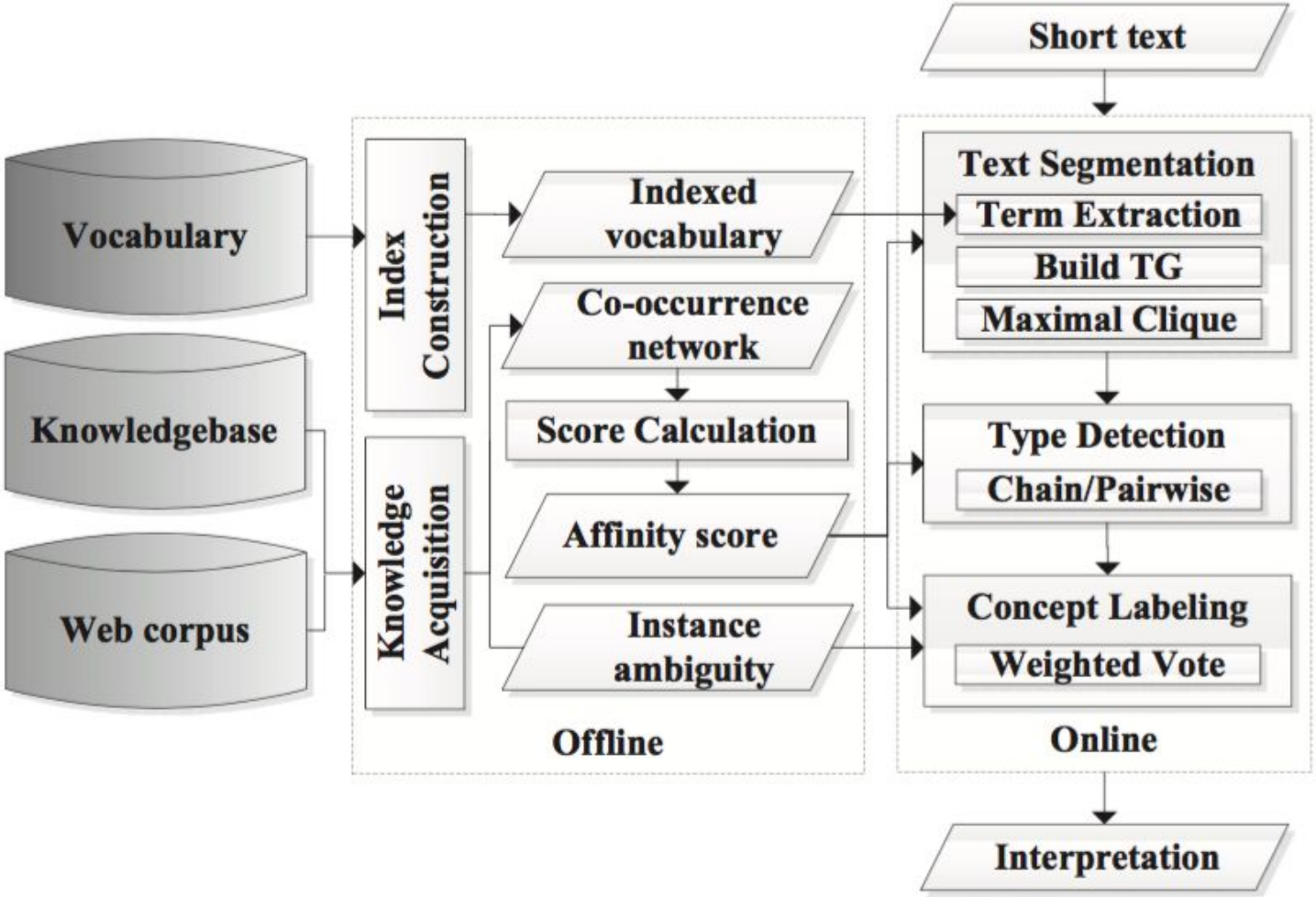
- **Co-occurence network** constructed from vocabulary, semantics

- Short text understanding divided into three subtasks:
    1. **Text sementation**
       - Term Graph (TG)
       - weighted Maximal Clique problem
       - randomized approximation algorithm for accuracy, efficiency

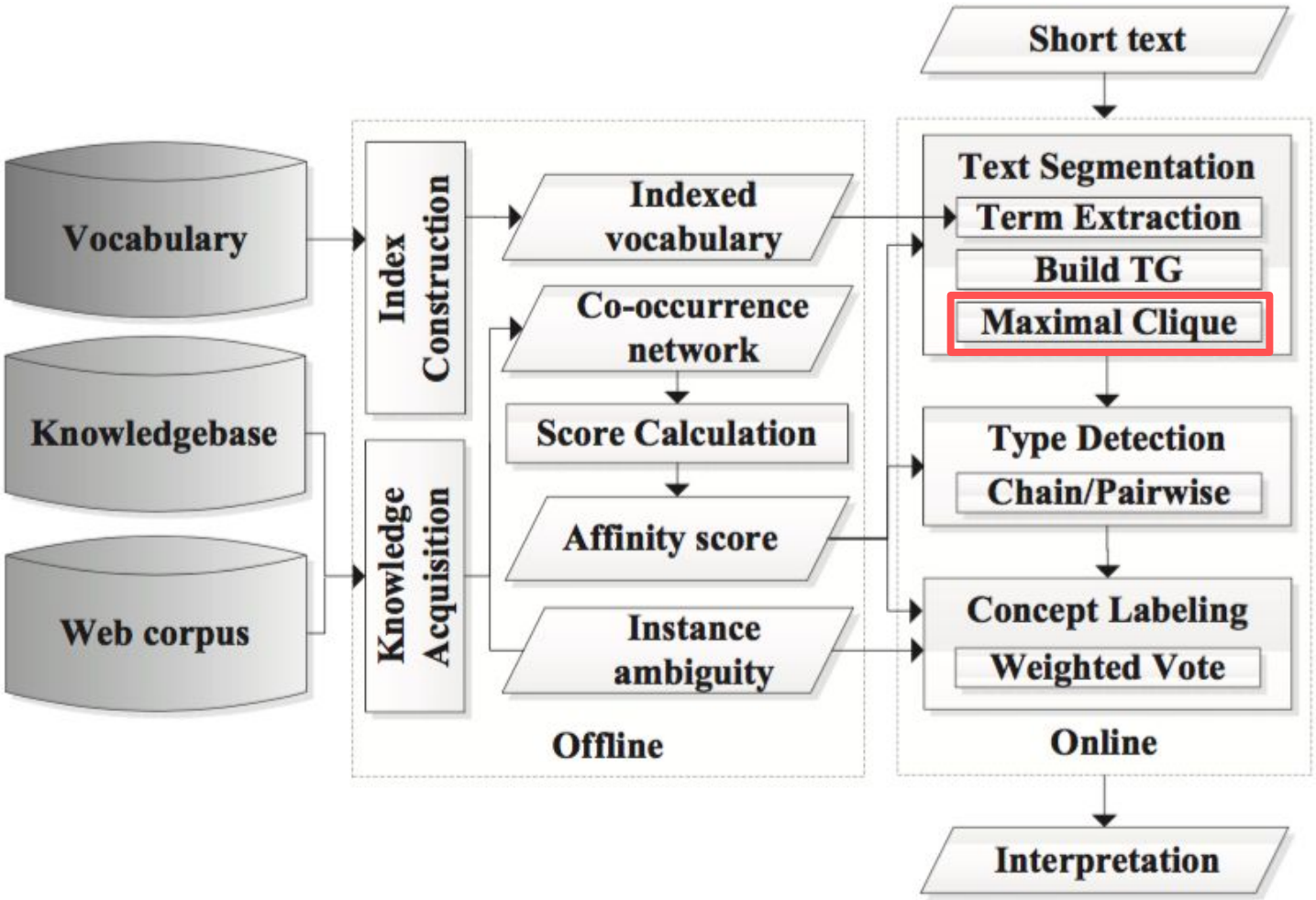
    2. **Type detection**
       - Chain Model, Pairwise Model (combine lexical, semantic features)

    3. **Concept labeling**
       - Weighted Vote algorithm to determine most appropriate semantics in cases of ambiguity

Methods


Short text
Vocabulary
Knowledgebase
Web corpus
Index Construction
Knowledge Acquisition
Indexed vocabulary
Co-occurrence network
Score Calculation
Affinity score
Instance ambiguity
Offline
Text Segmentation
Term Extraction
Build TG
Maximal Clique
Type Detection
Chain/Pairwise
Concept Labeling
Weighted Vote
Online
Interpretation

Methods


Short text
Vocabulary
Knowledgebase
Web corpus
Index Construction
Knowledge Acquisition
Indexed vocabulary
Co-occurrence network
Score Calculation
Affinity score
Instance ambiguity
Offline
Text Segmentation
Term Extraction
Build TG
Maximal Clique
Type Detection
Chain/Pairwise
Concept Labeling
Weighted Vote
Online
Interpretation

Methods: Text Segmentation

- Goal: find the **best segmentation** of a sequence of terms

- Use **Term Graph (TG)**
  - Nodes = candidate terms, weighted to represent their coverage of words within the short text
  - Edges = connect nodes when they are not mutually exclusive (e.g. “april in paris” and “april” are ME), weighted to reflect strength of mutual reinforcement (e.g. MR between “april in paris” and “lyrics”)

➛ Best segmentation is a **sub-graph of TG** s.t. the sub-graph:
  - Is a clique (complete graph; selected terms are not ME)
  - Has 100% word coverage
  - Has the largest average edge weight

- *Clique with 100% word coverage =* ***maximal clique of TG***

Methods: Maximal Clique

- Brute Force algorithm: too time-consuming

➛ Randomized algorithm: Maximal Clique by Monte Carlo

Algorithm 1 Maximal Clique by Monte Carlo (MaxCMC)

Input:
$G = (V, E)$; $W(E) = \{w(e) | e \in E\}$
Output:
$G' = (V', E')$; $s(G')$

```
V' = ∅; E' = ∅
while E ≠ ∅ do
    randomly select e = (u, v) from E with probability proportional to its weight
    V' = V' ∪ {u, v}; E' = E' ∪ {e}
    V = V − {u, v}; E = E − {e}
    for each t ∈ V do
        if e' = (u, t) ∉ E or e' = (v, t) ∉ E then
            V = V − {t}
            remove edges linked to t from E: E = E − {e' = (t, *)}
        end if
    end for
end while
calculate average edge weight: s(G') = Σ_{e∈E'} w(e) / |E'|
```

Algorithm 2 Chunking by Maximal Clique (CMaxC)

Input:
$G = (V, E)$; $W(E) = \{w(e) | e \in E\}$
number of times to run Algorithm 1: k
Output:
$G'_{best} = (V'_{best}, E'_{best})$

```
s_max = 0
for i = 1; i ≤ k; i++ do
    run Algorithm 1 with ¡G'_i = (V'_i, E'_i),s(G'_i)¿ as output
    if s(G'_i) > s_max then
        G'_best = G'_i; s_max = s(G'_i)
    end if
end for
```

Methods: MaxCMC

Algorithm 1 Maximal Clique by Monte Carlo (MaxCMC)

Input:
$G = (V, E)$; $W(E) = \{w(e) | e \in E\}$
Output:
$G' = (V', E')$; $s(G')$

```
V' = ∅; E' = ∅
while E ≠ ∅ do
    randomly select e = (u, v) from E with probability proportional to its weight
    V' = V' ∪ {u, v}; E' = E' ∪ {e}
    V = V − {u, v}; E = E − {e}
    for each t ∈ V do
        if e' = (u, t) ∉ E or e' = (v, t) ∉ E then
            V = V − {t}
            remove edges linked to t from E: E = E − {e' = (t, *)}
        end if
    end for
end while
calculate average edge weight: s(G') = Σ_{e∈E'} w(e) / |E'|
```

1. Randomly select edge e = (u,v)
   - *P*(e) proportional to weight of e

➛ *Output sub-graph G' is a maximal clique of the original TG.*

Methods: MaxCMC

Algorithm 1 Maximal Clique by Monte Carlo (MaxCMC)

Input:
$G = (V, E)$; $W(E) = \{w(e) | e \in E\}$
Output:
$G' = (V', E')$; $s(G')$

```
V' = ∅; E' = ∅
while E ≠ ∅ do
    randomly select e = (u, v) from E with probability proportional to its weight
    V' = V' ∪ {u, v}; E' = E' ∪ {e}
    V = V − {u, v}; E = E − {e}
    for each t ∈ V do
        if e' = (u, t) ∉ E or e' = (v, t) ∉ E then
            V = V − {t}
            remove edges linked to t from E: E = E − {e' = (t, *)}
        end if
    end for
end while
calculate average edge weight: s(G') = Σ_{e∈E'} w(e) / |E'|
```

1. Randomly select edge e = (u,v)
   - *P*(e) proportional to weight of e

2. a. Remove all nodes disconnected (i.e. ME) with u, v
   b. Remove all edges that are linked to the deleted nodes

➛ *Output sub-graph G' is a maximal clique of the original TG.*

Methods: MaxCMC

Algorithm 1 Maximal Clique by Monte Carlo (MaxCMC)

Input:
$G = (V, E)$; $W(E) = \{w(e) | e \in E\}$
Output:
$G' = (V', E')$; $s(G')$

```
V' = ∅; E' = ∅
while E ≠ ∅ do
    randomly select e = (u, v) from E with probability proportional to its weight
    V' = V' ∪ {u, v}; E' = E' ∪ {e}
    V = V − {u, v}; E = E − {e}
    for each t ∈ V do
        if e' = (u, t) ∉ E or e' = (v, t) ∉ E then
            V = V − {t}
            remove edges linked to t from E: E = E − {e' = (t, *)}
        end if
    end for
end while
calculate average edge weight: s(G') = Σ_{e∈E'} w(e) / |E'|
```

1. Randomly select edge e = (u,v)
   - *P*(e) proportional to weight of e
2. a. Remove all nodes disconnected (i.e. ME) with u, v
   b. Remove all edges that are linked to the deleted nodes
3. Repeated until no edges can be selected

➛ *Output sub-graph G' is a maximal clique of the original TG.*

Methods: MaxCMC

Algorithm 1 Maximal Clique by Monte Carlo (MaxCMC)

Input:
$G = (V, E)$; $W(E) = \{w(e) | e \in E\}$
Output:
$G' = (V', E')$; $s(G')$

```
V' = ∅; E' = ∅
while E ≠ ∅ do
    randomly select e = (u, v) from E with probability proportional to its weight
    V' = V' ∪ {u, v}; E' = E' ∪ {e}
    V = V − {u, v}; E = E − {e}
    for each t ∈ V do
        if e' = (u, t) ∉ E or e' = (v, t) ∉ E then
            V = V − {t}
            remove edges linked to t from E: E = E − {e' = (t, *)}
        end if
    end for
end while
calculate average edge weight: s(G') = Σ_{e∈E'} w(e) / |E'|
```

➛ *Output sub-graph G′ is a maximal clique of the original TG.*

1. Randomly select edge e = (u,v)
   - *P*(e) proportional to weight of e
2. a. Remove all nodes disconnected (i.e. ME) with u, v
   b. Remove all edges that are linked to the deleted nodes
3. Repeated until no edges can be selected
4. Evaluate G′ with score = average edge weight $\frac{\sum_{e \in E'} w(e)}{|E'|}$

Methods: CMaxC

Algorithm 2 Chunking by Maximal Clique (CMaxC)

Input:

$G = (V, E)$; $W(E) = \{w(e) | e \in E\}$

number of times to run Algorithm 1: k

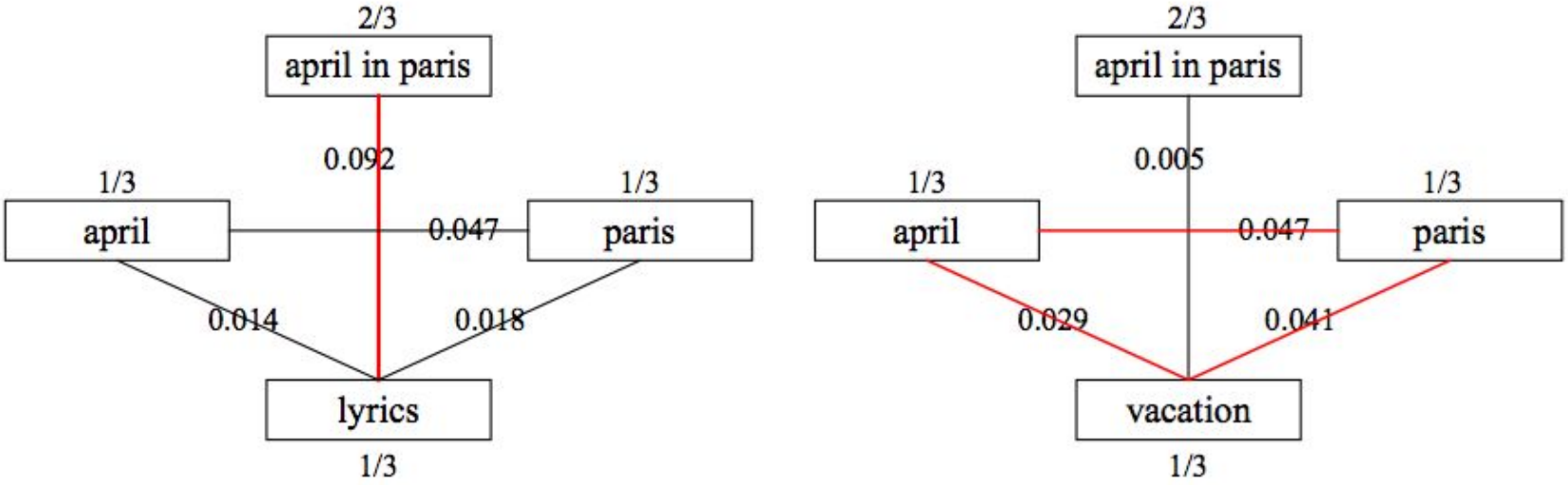
Output:

$G'_{best} = (V'_{best}, E'_{best})$

```
s_max = 0
for i = 1; i ≤ k; i++ do
    run Algorithm 1 with ¡G'_i = (V'_i, E'_i),s(G'_i)¿ as output
    if s(G'_i) > s_max then
        G'_best = G'_i; s_max = s(G'_i)
    end if
end for
```

1. Run MaxCMC *k* times to find best G′

Text Segmentation: Example



(a) A coherent segmentation of "april in paris lyrics" is {april in paris, lyrics}.

(b) A coherent segmentation of "vacation april in paris" is {vacation, april, paris}.

Results

Precision of text segmentation.

		Longest Cover	MaxCBF	MaxCMC
query	ambig	0.905	**0.980**	0.970
	general	0.972	**0.990**	0.982
	all	0.954	**0.984**	0.979
tweet		0.961	**0.980**	0.973

- MaxCMC > LC (semantics > surface features alone)
- MaxCMC ≅ MaxCBF
- MaxCMC 7.5% better for ambiguous queries, 1.8% better for general queries

Conclusions

- Semantic knowledge essential for effective, efficient understanding of short texts
- Randomized approximation algorithm for weighted Maximal Clique maintains accuracy while improving efficiency of text segmentation
- Future interests: NLP, analysis of growing corpora

Questions?

Thank you!

DOI 10.1109/TKDE.2016.2571687, IEEE Transactions on Knowledge and Data Engineering

Predicting Sex as a Soft-biometrics from Device Interaction Swipe Gestures

Oscar Miguel-Hurtado, Sarah V. Stevenage, Chris Bevan, Richard Guest

Jackie Byun
11/1/16

Motivation

- The predicted soft-biometrics traits can allow touchscreen computers-based systems to tailor their interaction to better suit the user’s characteristics.
- This information could also improve the performance of continuous authentication biometrics systems deployed in touchscreen devices by greatly decreasing search time in large databases.

Background

- Soft-biometrics: age, ethnicity, sex, height, weight, scars and tattoos, etc.
- Hard-biometrics: fingerprint, iris, and face

Machine Learning Classifiers

1) **Decision tree**: one of the most commonly used algorithms for automatic learning. The decision trees are composed of nodes (which test the value of an attribute), branches (path to follow based on the attribute value) and leaves (which provide the classification of the instance).
2) **Support vector machine (SVM)**: SVM algorithms are based on finding the optimal separating hyperplane that maximizes the margin, in other words, the hyperplane that gives the largest minimum distance to the training examples. These points are mapped so that the separate categories are divided by a clear gap that is as wide as possible.
    - Empirically good performance: successful applications in many fields (bioinformatics, text, image recognition,...)
3) **Multilinear logistic regression**: is one of the most commonly used tools for discrete data analysis. It is used to predict the probabilities of the different classes analysed given a set of independent variables. It represents a particular solution to the classification problem that assumes that a linear combination of the observed features can be used to determine the probability of each particular outcome of the dependent (aka categorical) variable.
4) **Naïve Bayes**: it is based on the probabilistic Bayes' rule and is particularly suited when the dimensionality of the inputs is high. In order to reduce the complexity of the high dimensionality, the naïve Bayes classifier assumes that the effect of the value of a particular feature on a given class is independent of the values of the other predictors.

Swipe Gesture Data Collection

Figure 1: Android OS application

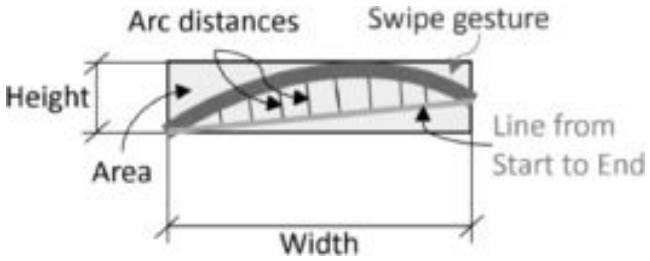


Figure 2: Swipe Feature Details

Table 1: Swipe Feature Set

#	Description	#	Description
1	Total length (px)	8	Maxima speed (px/ms)
2	Total time (ms)	9	Average speed (px/ms)
3	Width (px)	10	Maxima acceleration (px/ms^2)
4	Height (px)	11	Average acceleration (px/ms^2)
5	Area (px^2)	12	Average arc distance (px)
6	Average thickness (px)	13	Max arc distance (px)
7	Average pressure	14	Angle start to end (degrees)

Results

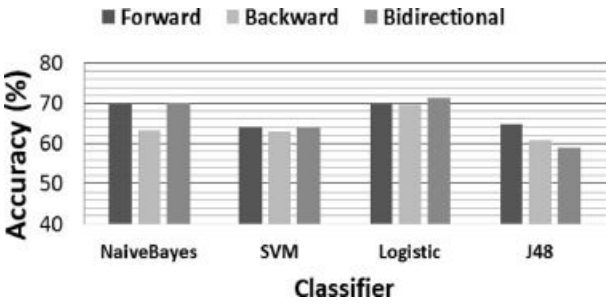


Figure 3: Down-to-up sex prediction score distribution.

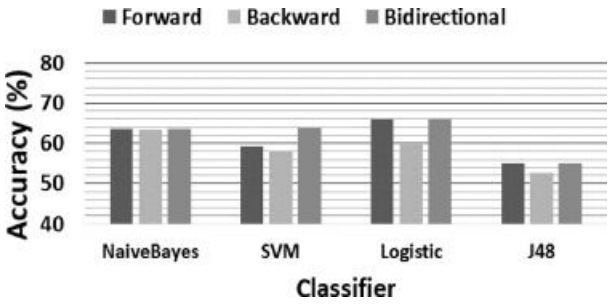


Figure 5: Right-to-left sex prediction score distribution.

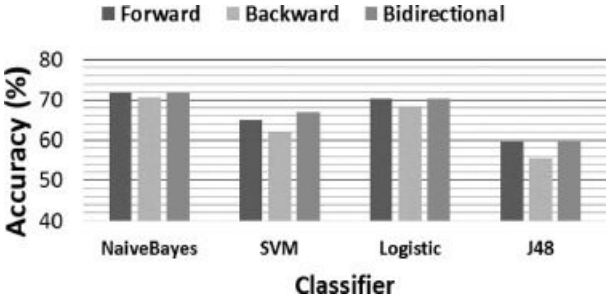


Figure 4: Left-to-right sex prediction score distribution.

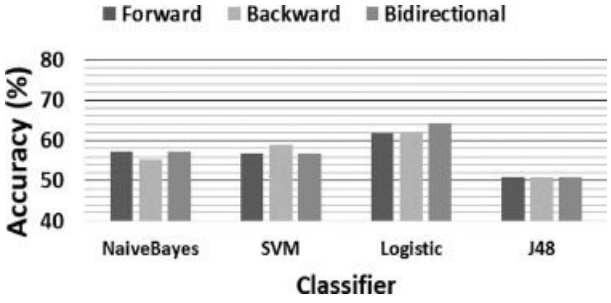


Figure 6: Up-to-down sex prediction score distribution.

Fusion Scheme

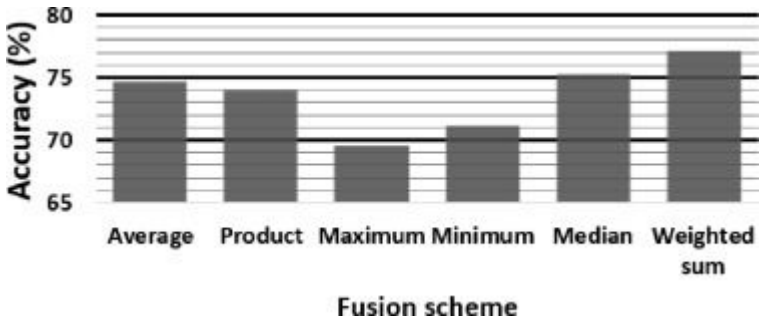


Figure 11: Matching score level fusion.
Up-to-down sex prediction score distribution.

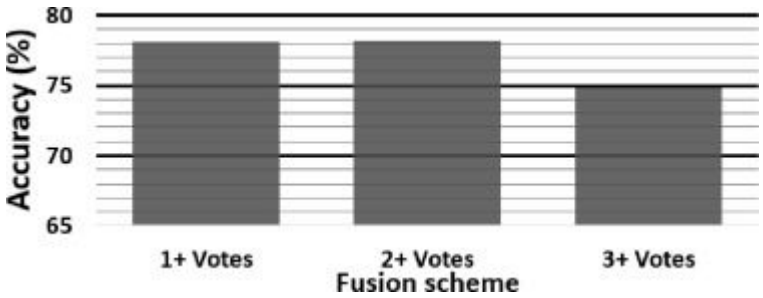


Figure 12: Decision level fusion. Up-to-down sex prediction score distribution.

Weighted sum of scores: $S_{fusion}=\sum_i w_i \cdot s_i$

$S_{fusion}=0.3 \cdot S_{UD}+0.7 \cdot S_{LR}$

Conclusions

- Confirmed the possibility of sex prediction from the swipe gesture data, obtaining an encouraging 71% accuracy from an individual swipe gesture direction.
- Best classifier: multilinear logistic regression
- The combination of direction swipe data enhanced the sex prediction accuracy by 6%, achieving a 78% accuracy rate using a decision voting scheme.

“Integrated patch model: A generative model for image categorization based on feature selection”

Li Chai Nov. 1

Background and Motivation

Key terms

- Low-level features
- Noisy features
- Salient feature

Methods

1. Feature selection strategy
2. IP model

Feature Selection Strategy

Salient Patch Selection

1. Detect salient patch
    - PCA -> a 15-PC vector
2. Construct visual keyword
    - K-means -> clusters
3. Remove similar or most non-common noises
    - ROD
    - Salient Entropy

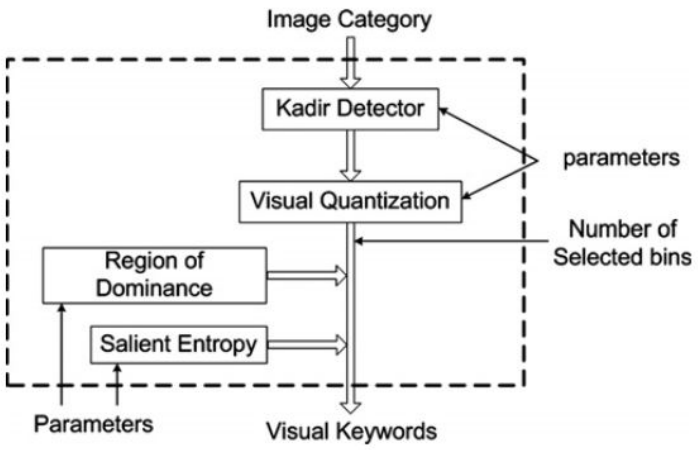


Fig. 2. Feature filter.

Feature Selection Strategy

Without noise reduction

After feature filter

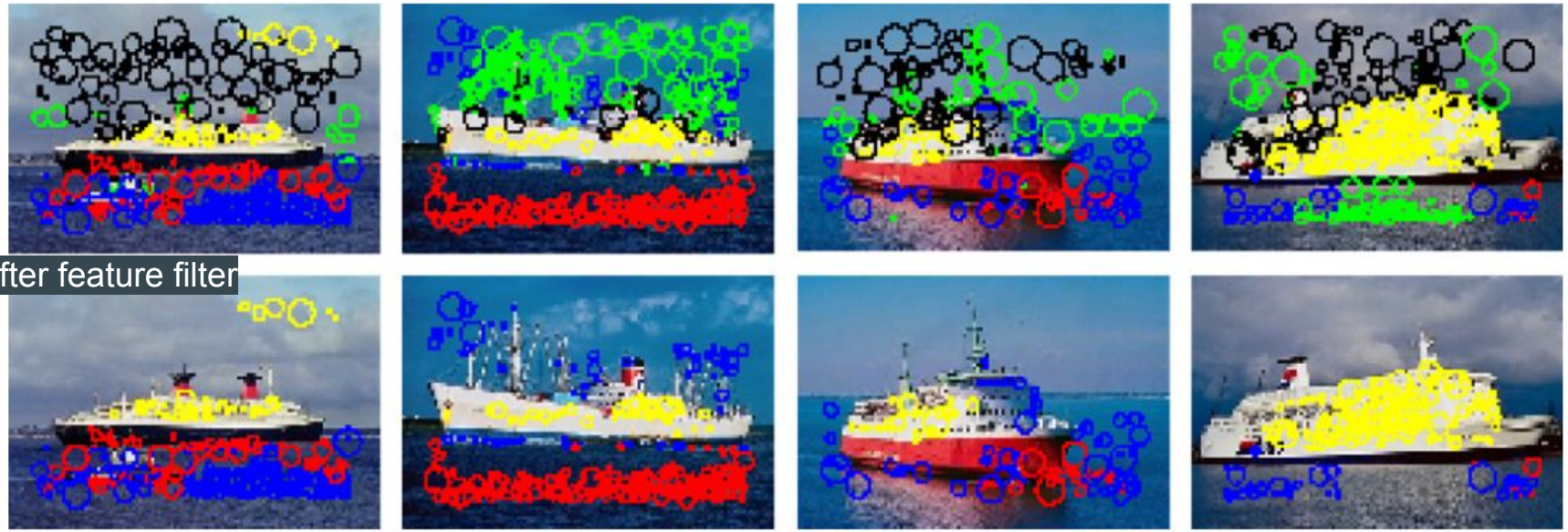


Fig. 3. Image examples with salient patches in “Ship” category in Corel image database.

Feature Selection Strategy

Salient Patch Selection

1. Detect salient patch
2. Construct visual keyword
3. Remove similar or most non-common noises

Feature Extraction → 64-dimensional feature for each patch

- 44 dimensional: auto-color correlogram
- 14 dimensional: color texture moments
- 6 dimensional: color moment

Image category description model

Feature Selection Strategy

Salient Patch Selection

1. Detect salient patch
2. Construct visual keyword
3. Remove similar or most non-common noises

Feature Extraction → 64-dimensional feature for each patch

- 44 dimensional: auto-color correlogram
- 14 dimensional: color texture moments
- 6 dimensional: color moment

Image category description model

- Multiply N cluster probabilities
- Use multiple mixture components to accommodate variabilities of same object
- Select representative patch for a visual keyword

Feature Selection Strategy

To test an image:

1. Calculate posterior probability in each category
    a. Detect salient patches and label as the nearest visual keyword
    b. Extract 64-dimensional feature
    c. Compute posterior prob of component in a category
2. Predict it with the highest prob category label

Results

- 5000 images from Corel image database
  a. 50 categories, each 100 images
- Within category, half randomly divided into training set and test set
- Select center of visual keywords as representative salient patch
- Evaluate performance of
  a. 35 image categories
  b. Diff. # of categories between training and test: 4:1
  c. Diff image # ratio

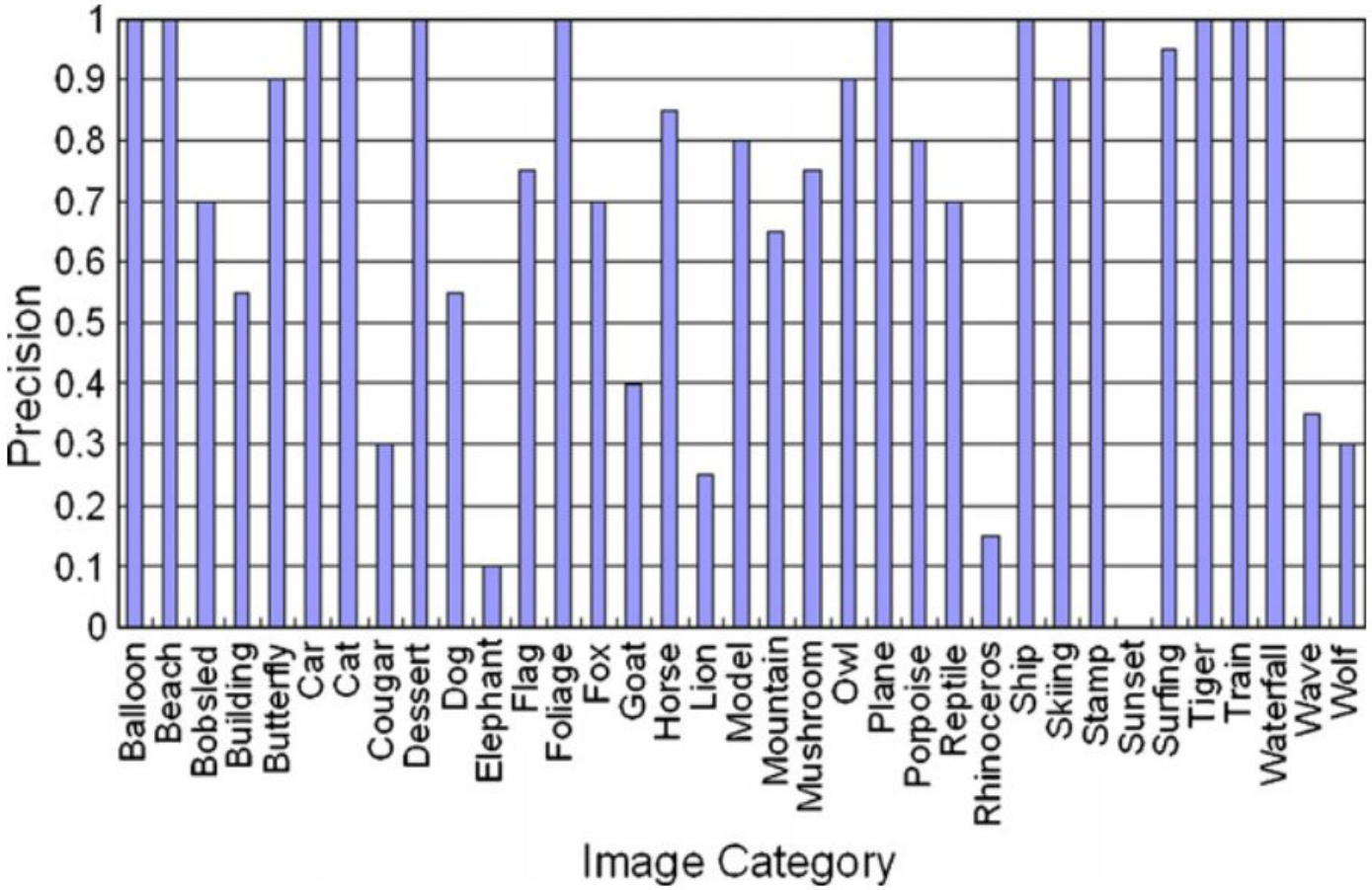


Fig. 6. Categorization precisions of 35 image categories.

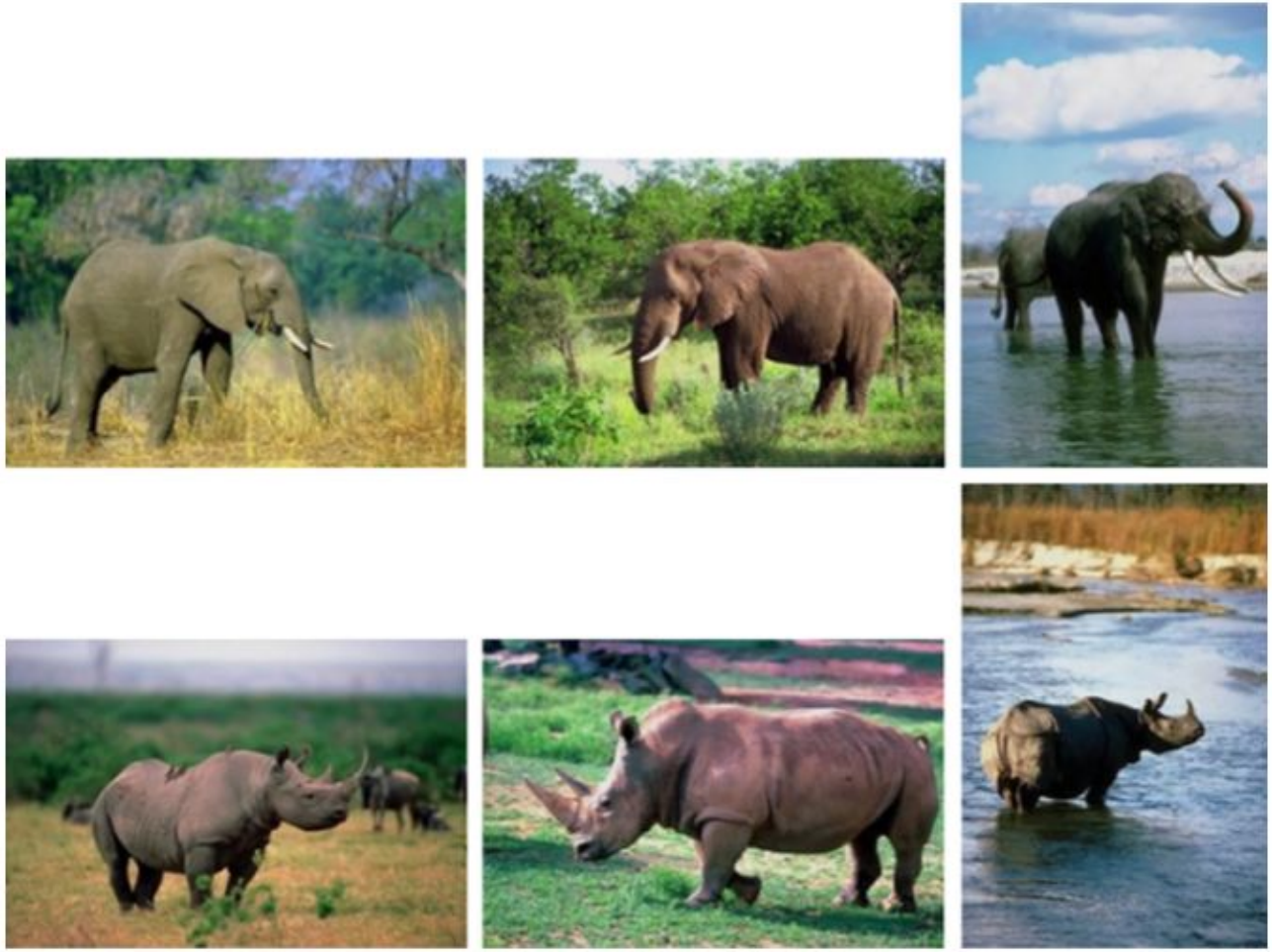


Fig. 7. Misclassified image examples for low categorization precisions (Elephant vs. Rhinoceros).

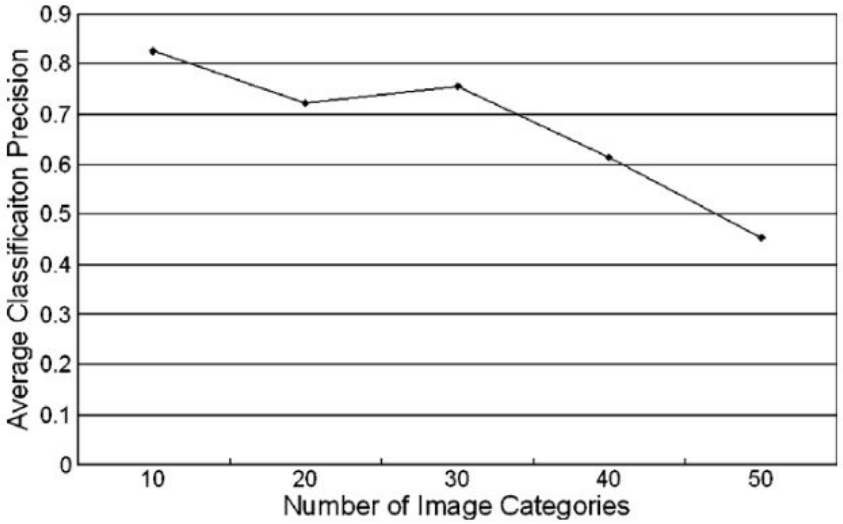


Fig. 8. Average categorization precisions in different numbers of image categories.

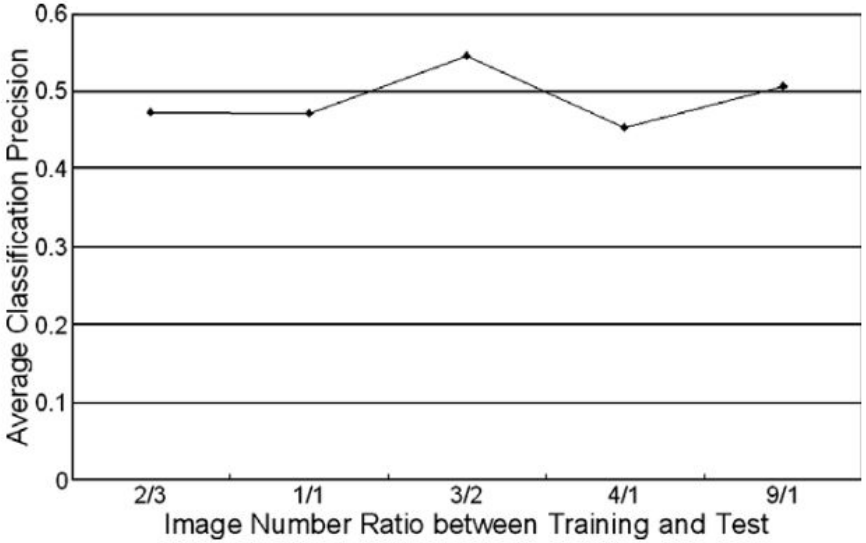


Fig. 9. Average categorization precisions in different image number ratios between training and test.

Conclusion

Lexicon-free Handwritten word spotting using character HHMs

Youyou Tian

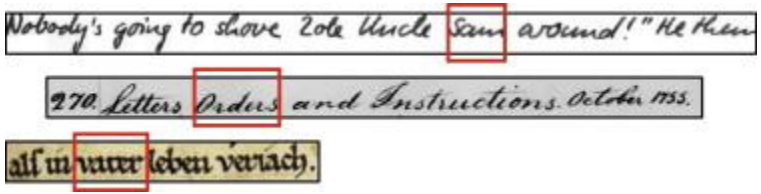
Motivation

- Handwriting is difficult to transcribe
  - Large vocabularies
  - Varying writing styles
- Automatic transcription is not very accurate

Motivation

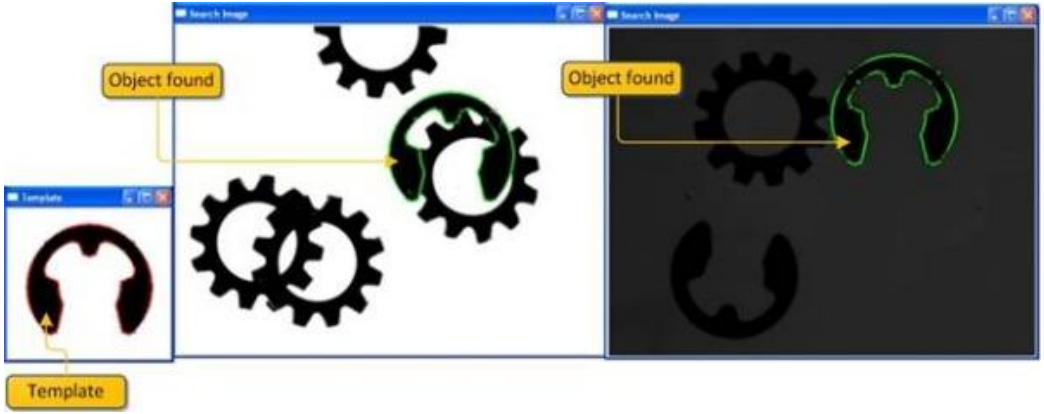
- Handwriting is difficult to transcribe
  - Large vocabularies
  - Varying writing styles
- Automatic transcription is not very accurate

- Solution: Handwritten word spotting
  - Retrieves keywords location in a document
- Many applications
  - Automatic postal mail sorting
  - Manuscript Preservation



Background

- Template-Based method
  - Matches a query word based on labeled (whole) keyword template images
  - There needs to be a template for each word to be spotted
  - Unknown vocabulary may not be spotted



http://www.codeproject.com/KB/graphics/Edge_Based_template_match/GeoMatch001.jpg

Background

- Template-Based method
  - Matches a query word based on labeled (whole) keyword template images
  - There needs to be a template for each word to be spotted
  - Unknown vocabulary may not be spotted

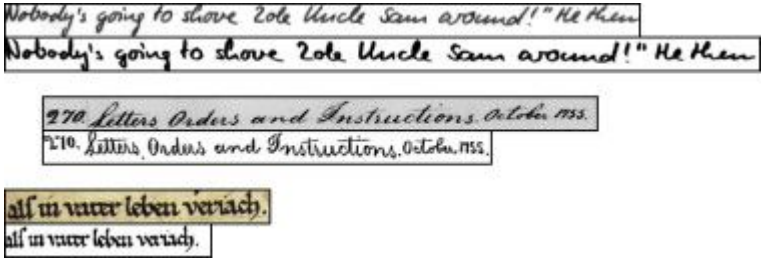
- Learning-based methods use HMM
  - Train a keyword model to score likeliness of a match
  - Needs a lot of training data
  - Past research was done on a word level

Solution

- Creating a Character based learning approach
- Only need a small amount of character samples compared to word combinations
  - Can spot arbitrary words
- However, set of transcribed text training images hard to obtain (historical languages)

Methods

- Image Processing
  - Normalize the data
  - Distort the image to remove skew and slant of the words

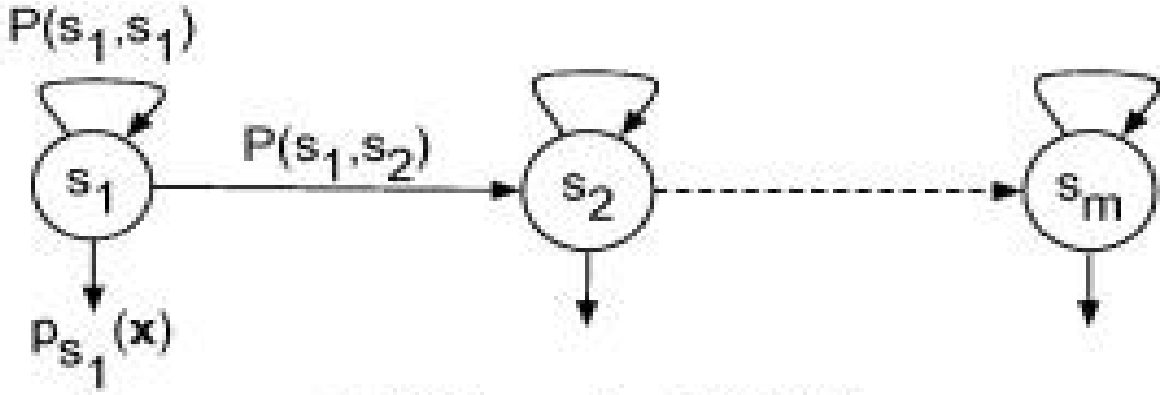


Methods

- Feature Extraction
  - 9 features based on previous work

Methods

- Hidden Markov Models
  - Can cope with characters connected with cursive

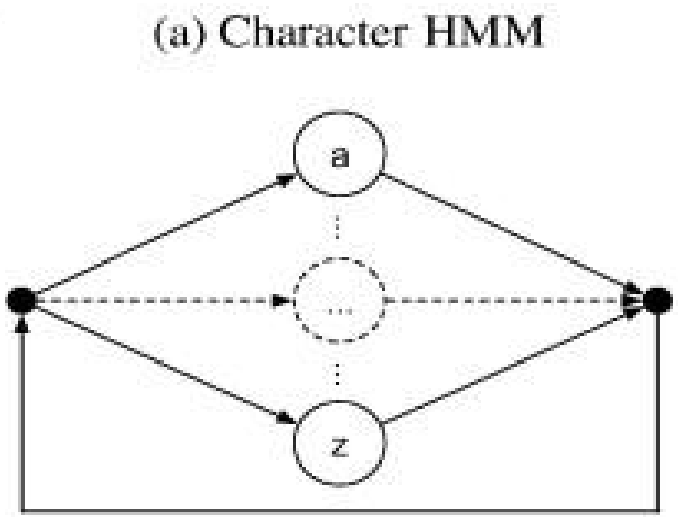


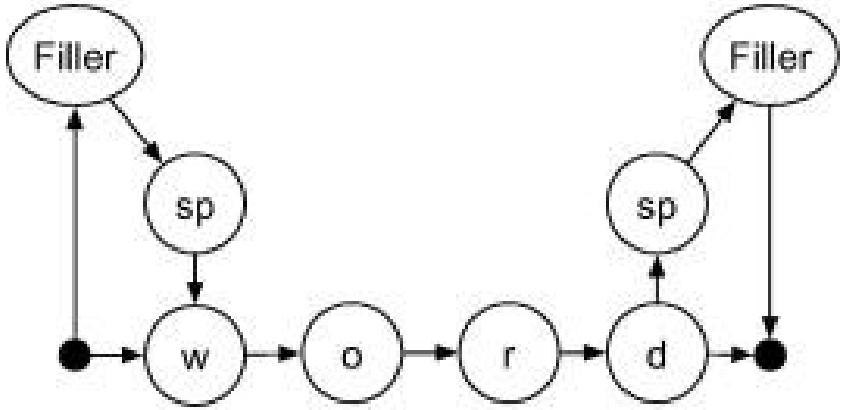
(a) Character HMM

$$p_{s_j}(\mathbf{x}) = \sum_{k=1}^{G} w_{jk} \mathcal{N}(\mathbf{x} | \mu_{jk}, \Sigma_{jk})$$

Methods

- Hidden Markov Models
  - Can describe a word as a sequence





(c) Keyword HMM

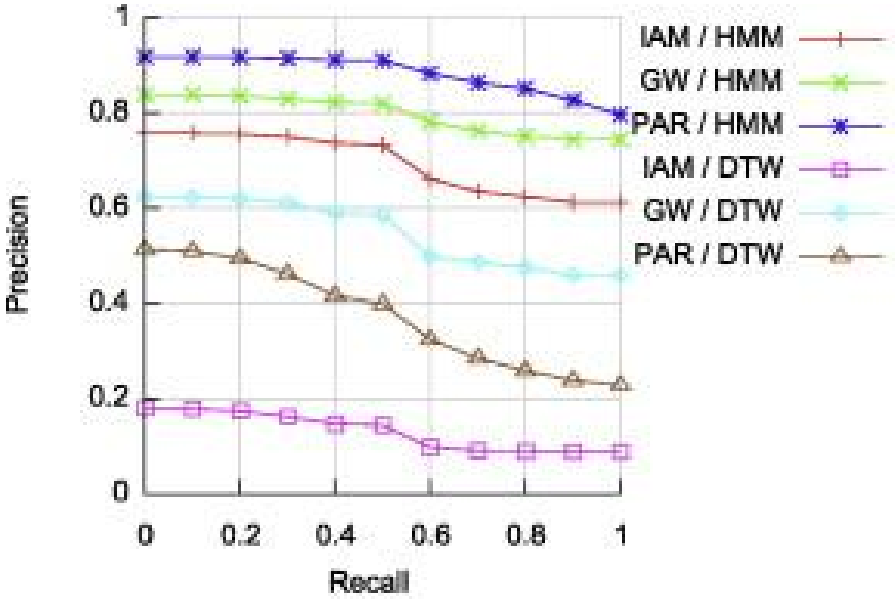
Scoring

- Use Bayes rule to get scoring
  -

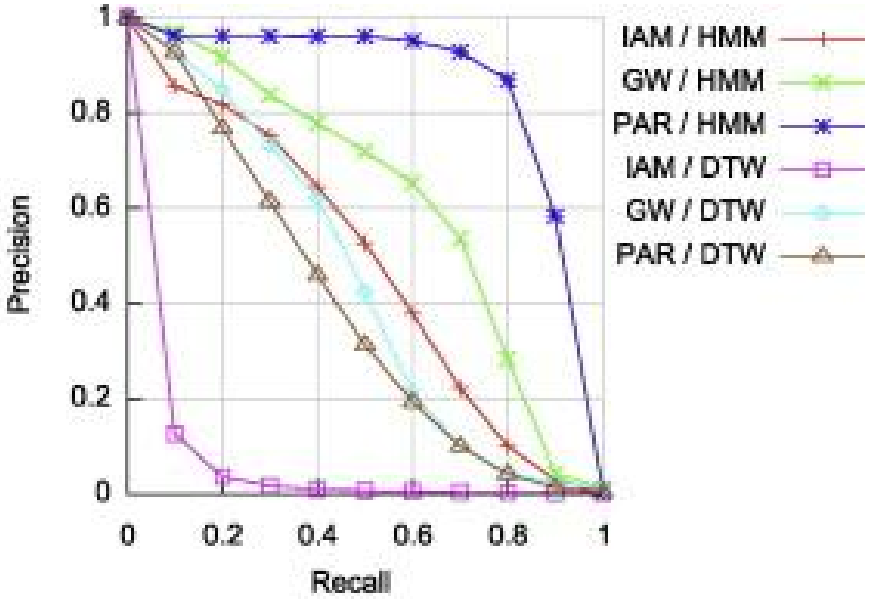
$$\log p(W|X_{a,b}) = \log p(X_{a,b}|W) + \log p(W) - \log p(X_{a,b})$$

$$s(X,W) = \frac{\log p(X|K) - \log p(X|F)}{L_K} \geqslant T$$

Results

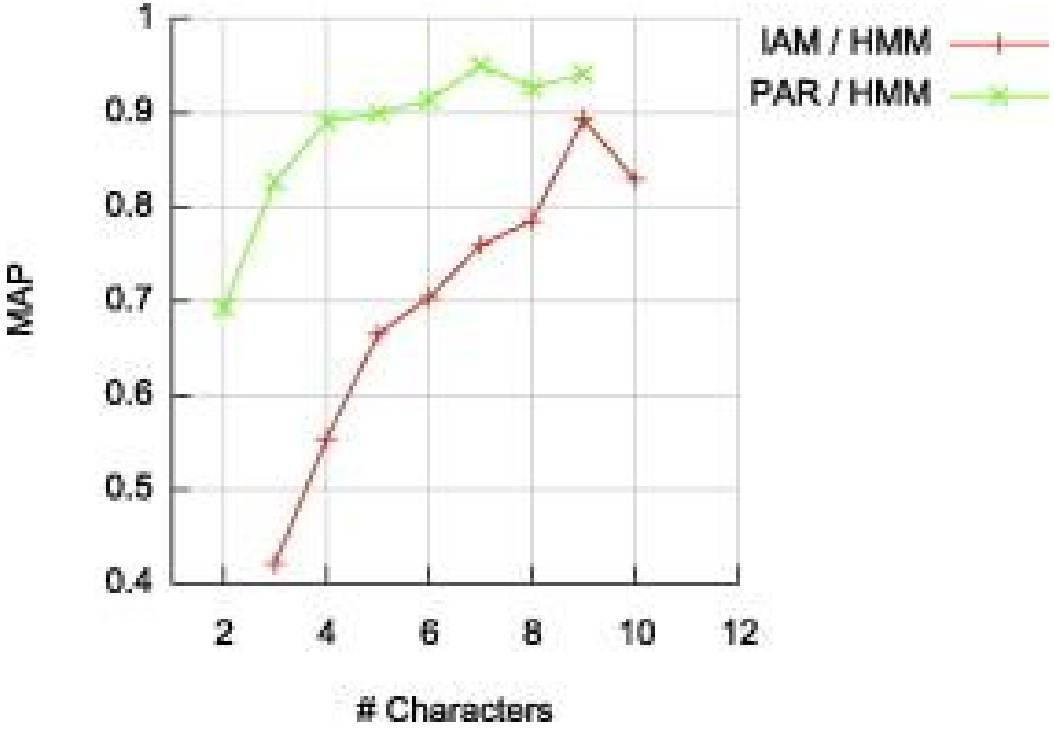


(a) Local Thresholds



(b) Global Threshold

Results



Conclusions

- This approach works well with the IAM and George Washington Database
- Limitations given the whole alphabet is required. Tests were done in english
- Possible Semi-structured approach