A Computer System for Automatically Identifying Text Structure in Writing

Laurence Anthony and George V. Lashkia

Dept. of Computer Science, Faculty of Engineering

Okayama Univ. of Science, 1-1 Ridai-cho, Okayama

anthony@ice.ous.ac.jp lashkia@ice.ous.ac.jp

http://antpc1.ice.ous.ac.jp

Presentation Outline

- Background
- Research Aim
- System Design
- Application to Research Abstracts
- Results
- Conclusions

Background

- Importance of Text Structure
  - Swales (1981, 1990), Carrell (1982)
    Hinds (1982, 1983), Hoey (1994), Winter (1994)
- Studies on Text Structure
  - **TITLES** - Dudley-Evans (1994), Anthony (2001)
  - **ABSTRACTS** - Ayers (1993), Posteguillo (1996)
  - **INSTRODUCTIONS** - Swales (1990), Anthony (1999)
  - **DISCUSSIONS** - Hopkins & Dudley-Evans (1988)
  - **PATENTS** - Bazerman (1994)
  - **GRANT PROPOSALS** - Connor & Mauranen (1999)
  - **LEGAL WRITING** - Bhatia (1993)

Background

- Problems with Analyzing Text Structure
  - A large corpus of text data
    (The text data must 'ACURATELY' represent what we hope to study)
  - A lot of research time
    (Time to analyze a lot of texts)
  - Good validation and reliability tests
- Most Text Structure Studies are 'Small Scale'

Background

- Swales (1981: p.13)

  "In effect, the discourse analyst labels something as X and then begins to see X occurring all over the place"

Background

- Henry et al. (2001)
  - 40 Application Letters
- Tarone et al. (2000)
  - 2 Physics Research Articles
- Connor et al. (1999)
  - 34 Grant Proposals
- Williams (1999)
  - 5 Medical Research Articles
- Anthony (1999)
  - 12 Computer Science Research Article Introductions

Research Aim

- Develop a Computer System to Process and Analyze Text Structure Automatically
  - A *'Learning System'* for text structure
    - Easy to Process a Large Corpus of Text Data
    - Fast
    - The analytic process is clearly defined
    - Easy to test the reliability and validity

System Design

- ‘Unsupervised Learning’ vs..
  ‘Supervised Learning’?
- In Unsupervised Learning,
  - Give the system text examples
  - Tell the system what ‘features’ to look at
  - Let the system find a model (set of classes) by defining a relation between the features and the examples
  - Classify new text examples by comparison with features in each class

Unsupervised Learning

- Give the system text examples
  - Text 1: Once upon a time, there was a ugly duckling.
  - Text 2: It lived on a lake.
  - Text 3: One day, the little bird turned into a swan.
  - Text 4: It lived happily, ever, after.
- Tell the system what ‘features’ to look at
  - All words except articles, No punctuation
- Define a relation between features and examples
  - Class 1 - once, upon, time, there, was, ugly, duckling
  - Class 2 - one, day, little, bird, turned, into, swan
  - Class 3 - it, lived, on, lake, happily, ever, after

Unsupervised Learning

- Unsupervised learning system models often DO NOT match our models
- Classify new text examples
  - Once upon a time, there were 3 bears (BEG)
  - The 3 bears lived in a big house. (MID)
  - They all stayed in the house happily ever after. (END)
- The system will decide ...
  - Class 1 (matching 'once', 'upon', 'time', 'there')
  - Class 3 (matching 'lived')
  - Class 3 (matching 'happily', 'ever', 'after')

System Design

- ‘Unsupervised Learning’ vs. ‘Supervised Learning’?
- In Supervised Learning,
  - Give the system a structure model (set of classes)
  - Give the system examples of the model
  - Tell the system what ‘features’ to look at
  - Define a relation between the classes and the features
  - Classify new text examples by comparing its features with those in each class

Supervised Learning

- Give the system a structure model (set of classes)
  - Class 1: BEGINNING
  - Class 2: MIDDLE
  - Class 3: END
- Give the system examples of the model
  - BEG: Once upon a time, there was a ugly duckling.
  - MID: It lived on a lake.
  - MID: One day, the little bird turned into a swan.
  - END: It lived happily, ever, after.
- Tell the system what ‘features’ to look at
  - All words except articles..., No punctuation

Supervised Learning

- Define a relation between classes and features
  - Class 1 (BEG) - once, upon, time, there, was, ugly, duckling
  - Class 2 (MID) - it, lived, on, lake, one, day, little, bird, turned, into, swan
  - Class 3 (END) - lived, happily, ever, after
- Classify new text examples
  - Once upon a time, there were 3 bears (BEG)
  - The 3 bears lived in a big house. (MID)
  - They all lived in the house happily ever after. (END)
- The system will decide…
  - Class 1 (BEG) (matching 'once', 'upon', 'time', 'there')
  - Class 2 (MID) (matching 'lived')
  - Class 3 (END) (matching 'lived', 'happily', 'ever', 'after')

Supervised Learning

- Problems
  - We need a ‘good’ model of structure
    - – But there are many models of structure in the literature
  - We need a set of ‘labeled examples’
    - – But many systems work well with only a few labeled examples
  - We need a ‘good’ set of features
    - – But language contains a LOT of redundant words! (e.g. a, the, of, in, ….)
    - – Building a list of features by hand is infeasible
  - We need a ‘good’ relation between the classes and the features???
    - – In practice, very simple relationships are effective!

Application of System to Research Abstracts

- Give the system a structure model:
  'Modified' CARS Model (Swales, 1990: Anthony, 1999)

Move 1 Establishing a Territory	**1.1**	**Claiming centrality**
	1.2	**Making topic generalizations**
	1.3	**Reviewing items of previous research**
Move 2 Establishing a niche	**2.1A**	**Counter claming**
	2.1B	**Indicating a gap**
	2.1C	**Question raising**
	2.1D	**Continuing a tradition**
Move 3 Occupying the niche	**3.1A**	**Outlining purpose**
	3.1B	**Announcing present research**
	3.2	**Announcing principal findings**
	3.3	**Evaluation of research**
	3.4	**Indicating RA structure**

Application of System to Research Abstracts

- Give the system examples of the model
  - 100 Abstracts (IEEE Trans. on PDS) divided into 692 labeled 'Steps Units' (only examples from 6 classes)
  - 554 Step Units (80%) used for 'training' the system
  - 138 Step Units (20%) used for 'testing' the system
- Tell the system what 'features' to look at
  - All word clusters up to 5 words long
  - Position of step unit in abstract (i.e. 1st, 2nd, 3rd, …)
- (Reduce 'Noise' in Features)
  - Rank words by 'importance' using:
    raw frequency, information gain
  - Use only high ranked words

Application of System to Research Abstracts

- There were many people in the park.
  - 1 word
    - there/ were/ many/ people/ in/ the/ park
  - 2 words
    - there were/ were many / many people/ people in / in the / the park
  - 3 words
    - there were many / were many people/ many people in / people in the / in the park
  - ...

Application of System to Research Abstracts

- There were many people in the park.
  - 1 word
    - there/ were/ many/ people/ in/ the/ park
  - 2 words
    - there were/ were many / many people/ people in / in the / the park
  - 3 words
    - there were many / were many people/ many people in / people in the / in the park
  - …

Application of System to Research Abstracts

- Define a relation between features and model
  - Use probability of words (features) being in each class

    Probability = Frequency of Word/Total number of words

    - Class 1 (Claiming Centrality)
      Class 2 (Making topic generalizations)
      Class 3 (Indicating a gap)
      Class 4 (Outlining purpose)
      Class 5 (Announcing principal findings)
      Class 6 (Evaluation of research)

Class 1:	Word 1 prob	Word 2 prob	Word 3 prob. …
Class 2:	Word 1 prob	Word 2 prob	Word 3 prob. …
Class 3:	Word 1 prob	Word 2 prob	Word 3 prob. …
Class 4:	Word 1 prob	Word 2 prob	Word 3 prob. …
Class 5:	Word 1 prob	Word 2 prob	Word 3 prob. …
Class 6:	Word 1 prob	Word 2 prob	Word 3 prob. …

Application of System to Research Abstracts

- Classify new text examples
  - For each new text, choose class with highest probability of having words (features)

    e.g. New Text only has features 3, 8, 10
  - Class 1 P= p.class 1 x p. f3 x p. f8 x p. f10 = 1.5
    Class 2 P= p.class 2 x ... = 1.8
    Class 3 P= p.class 3 x ... = 2.7
    Class 4 P= p.class 4 x ... = 2.3
    Class 5 P= p.class 5 x ... = 1.8
    Class 6 P= p.class 6 x ... = 1.2
  - Choose Class 3

Results

- Classification Accuracy (Overall)
  - 554 Step Units used for ‘training’ the system
  - 138 Step Units used for ‘testing’ the system

No. of Features	Accuracy (Raw Frequency)	Accuracy (Information Gain)
2208 (all)	56 %	-
1000	51 %	70 %
700	56 %	70 %
500	59 %	69 %
300	59 %	69 %
100	54 %	-

Note: Random guessing has an accuracy of 16.66% (NOT 50%!)
Choosing the most common class = 26%

Results

- Classification Accuracy (Each Step Unit)
  - Number of features = 700
  - Ranked by information gain measure
  - Accuracy (overall) = 70%

Class	Step 1.1	Step 1.2	Step 2.1b	Step 3.1b	Step 3.2	Step 3.3
Step 1.1	2 (43 %)	4	0	0	1	0
Step 1.2	0	17 (77 %)	0	0	4	1
Step 2.1b	0	2	1 (17 %)	0	2	1
Step 3.1b	0	0	0	34 (92 %)	3	0
Step 3.2	0	2	0	2	25 (66 %)	9
Step 3.3	0	1	0	2	8	17 (61 %)

Note: Classifications correspond with CARS Model 'moves'
(Accuracy=88% when using 'second opinion')
The system makes the same mistakes as humans.

Results

- Classification Accuracy
  - (For different data sets)
  - Number of features = 700 (Ranked by information gain)
    - Data Set 1 Accuracy = 70%
    - Data Set 2 Accuracy = 69%
    - Data Set 3 Accuracy = 69%
- Classification Accuracy
  - (For different data sets)
  - (Using 1st and 2nd ranked classification - 'Second Opinion')
  - Number of features = 700 (Ranked by information gain)
    - Data Set 1 Accuracy = 88%
    - Data Set 2 Accuracy = 86%
    - Data Set 3 Accuracy = 86%

Results

- A ‘Windows’ Interface
  - To enable researchers to use the system it needs to be easily accessible via a ‘windows’ interface
  - A ‘windows’ system has been built using the programming language PERL 5.6 and PERL/Tk
    - The system offers suggestions about the structure of new texts
    - The structure suggestions can be edited/corrected
    - The new texts can be added to the database of training example texts
    - The system can ‘relearn’ the structure and improve over time

Conclusions

- A computer system was developed to analyze text structure
  - Learning method: ‘Supervised Learning’
  - Training examples: 554
  - Testing example: 138
  - Accuracy 70% (88% when using second opinion)
- System errors are similar to those made by humans
- The accuracy needs to be improved
  - Currently working on better feature selection

Conclusions

- The system runs in a ‘windows’ environment
- The system offers ‘suggestions’ which can be edited by the user
- The ‘windows’ interface needs to be enhanced
  - I hope to make a complete environment to help researchers solve many ‘supervised learning’ problems
    - Move analysis, Text categorization, Author authenticity etc.

A Computer System for Automatically Identifying Text Structure in Writing

Laurence Anthony and George V. Lashkia

Dept. of Computer Science, Faculty of Engineering

Okayama Univ. of Science, 1-1 Ridai-cho, Okayama

anthony@ice.ous.ac.jp lashkia@ice.ous.ac.jp

http://antpc1.ice.ous.ac.jp