Lecture 1. Text Pre-processing

Definations:
- **Words**: Sequence of characters with a meaning and/or function
- **Sentence**: “The student is enrolled at the University of Melbourne.”
- **Word token**: each instance of “the” in the sentence above.
- **Word type**: the distinct word “the”.
- **Lexicon**: a group of word types.
- **Document**: one or more sentences.
- **Corpus**: a collection of documents.

Text Normalisation
- Remove unwanted formatting (e.g.HTML)
- Segment structure(e.g.sentences)
- Tokenise words
- Normalise words
- Remove unwanted words

Word normalisation
- Lowercasing(Australia->australia)
- Removing morphology (Inflectional and Derivational)
- Correcting spelling
- Expanding abbreviations

Inflectional Morphology -> grammatical variants
Derivational morphology -> distinct words (suffixes and prefixes)

Lemmatisation:
Removing any inflection to reach the uninflected form, the lemma.

Stemming:
Stemming strips off all suffixes, leaving a stem.

Lecture 2 Information Retrieval: the Vector space model

Evaluation on Test collections:
IR research often use reusable test collections constructed for reproducible IR evaluation.

Document representation
Bag-of-Word (BOW) -> term-document matrix

Cosine distance
$$Cos\,(a, b) \;=\; \frac{a \cdot b}{|a| \times |b|}$$

TF-IDF
TF: The number of the term occurs in a specific document.
IDF: The number of document that the term occurs.

Query processing in VSM
Normalise TF-IDF square root -> divide each item of rows
Similarity: Doc dot Query

BM25

$$\begin{aligned} w_{td} = & \left[\log \frac{N - f_t + 0.5}{f_t + 0.5}\right] && \text{(idf)} \\ & \times \frac{(k_1 + 1) f_{d,t}}{k_1\left((1 - b) + b\frac{L_d}{L_{ave}}\right) + f_{d,t}} && \text{(tf and doc. length)} \\ & \times \frac{(k_3 + 1)\, f_{q,t}}{k_3 + f_{q,t}} && \text{(query tf)} \end{aligned}$$

Term-wise processing
With the query as a pseudo-document need only dot-product consider terms that are in the query and in the document.

Inverted Index
- Terms as rows
- Values as lists of (docID, weight) pairs, aka posting list
- (weights listed are the normalised TF*IDF values)

Lecture 3: Index compression and efficient query processing

Entropy: Information content of a text T

$$H(T) = -\sum_{s \in \Sigma} \frac{f_s}{n} \log_2 \frac{f_s}{n}$$

where fs is the frequency of symbol s in T and n is the length of T.

Gaps:
Gaps between two adjacent integers can be much smaller

Variable Byte Compression:
Use variable number of bytes to represent integers. Each byte contains 7 bits “payload” and one continuation bit.

Fast Searching

1. Binary search over uncompressed sample values to find destination block
2. Decompress destination block to determine final offset in postings list

Top-k Retrieval
Retrieve the top k items for a given query without having to evaluate all documents.

The WAND Algorithm

- Keep track of the top-k highest scored documents.
- For each unique term in the collection store the maximum contribution it can have to any document score in the collection.
- Skip over documents that can not enter the top-k highest results.

Maximum contribution
The Maximum contribution of a term q as the largest score any document in the collection can have for the query Q only consisting of q.

Lecture 4: Query Completion and Expansion

High Level Algorithm:

1. Retrieve set of candidates
2. Rank candidates by frequency
3. Re-rank highest ranked candidates and return top-k

Completion Types:

- Prefix match.
- Substring match.
- Multi-term prefix match.
- Relaxed match.

Prefix match (Trie+RMQ based Index)

Store array with frequencies corresponding to each query. Subtree corresponds to range in frequency array. Find the top-K highest numbers in that range.

Range Maximum Queries

Given an array A of n numbers, and a range [l, r] of size m, find the positions of the K largest numbers in A[l, r].

1. Find position of largest element of A[i,j].
2. Recurse to A[i,p−1] and A[p+1,j].
3. Keep going until you have the K largest elements.
4. Runtime O(K log K).
5. Instead of precomputing all O(n2) ranges A[i, j], for each position A[i], precompute only log n ranges of increasing size: A[i, i + 1],A[i, i + 2],A[i, i + 4],A[i, i + 8].
6. Any range A[l, r] can be decomposed into two ranges A[l, Y] and A[Z, r] where Y = l + 2x and Z = r − 2y such that Z ≥ l, Y ≤ r and, A[l, Y], A[Z, r] overlap. Then, RMQ(A[i, j]) = max(RMQ(A[l, Y]), RMQ(A[Z, r]))
7. Total space cost O(n log n).

Query Expansion

- User and documents may refer to a concept using different words
- Vocabulary mismatch
- Users often attempt to fix this problem manually
- Adding these synonyms should improve query performance

Global Query Expansion

Retrieve synonyms from WordNet and Word2Vec.

User relevance feedback

Relevance Feedback. User provides feedback to the search engine by indicating which results are relevant.

Pseudorelevance feedback
- Take top-K results of original query
- Determine important/informative terms/topics (topic
- modelling!) shared by those documents Expand query by those terms
- No explicit user feedback needed (also called blind relevance feedback)

Indirect relevance feedback
For a query look at what users click on in the result page

Lecture 5: Index Construction and Advanced Queries

Static construction
- Invert one batch
- Merge batches

Auxiliary Index
- One static large static index on disk.
- As new documents arrive keep them in-memory in second index.

Logarithmic Index
- Store index of size $2^i \times n$
- Construction cost: $N \log(N/n)$

Phrase queries
- Inverted Index based (Positional Inverted Index)
- String matching indexes (Suffix Arrays)

More advanced queries
- Wildcard/misspelling queries (Sydney vs. Sidney: query S?dney)
- Regular expression queries (“[Jj]john.*”@smith.com???”)
- Proximity queries (“president” close to “obama”)

Lecture 6: IR Evaluation and re-ranking

Hard to characterise the quality of a system's results *

- a subjective problem, depends on the user's information need and how well the results meet that need.
- query is not the information need itself, but an expression thereof.

Simplifying assumptions

- Retrieval is ad-hoc
- Effectiveness based on relevance

recall is hard to calculate

Precision-oriented metrics

- Precision@K: compute precision using only ranks 1 .. k
- Average Precision (AP): take average over precision@k for each k where rank k item is relevant; measure becomes rank sensitive
- Mean Average Precision (MAP): AP averaged across multiple queries

Rank-biased precision

$$RBP = (1 - p) \times \sum_{i=1}^{d} r_i \times p^{i-1}$$

Re-ranking

- Use BM25 as a first step in multi-stage retrieval system
- Use complex trained ranking model store rank the original BM25 ranking

Rank objective

- Point-wise objective
- Pair-wise objective

Lecture 7: Text Classification

Text classification tasks

- Topic classification
- Sentiment analysis
- Authorship attribution
- Native-language identification
- Automatic fact-checking

Building a Text classifier

- Identify a task of interest
- Collect an appropriate corpus
- Carry out annotation
- Select features
- Choose a machine learning algorithm
- Tune hyperparameters using held-out development data
- Repeat earlier steps as needed
- Train final model
- Evaluate model on held-out test data

Naïve Bayes

- Finds the class with the highest likelihood under Bayes law.
- Naïvely assumes features are independent.
- Pros:
    - Fast to “train” and classify;
    - robust, low- variance;
    - good for low data situations;
    - optimal classifier if independence assumption is correct;
    - extremely simple to implement.
- Cons:
    - Independence assumption rarely holds;
    - low accuracy compared to similar methods in most situations;
    - smoothing required for unseen class/feature combinations

Logistic Regression

- A linear model, but uses softmax “squashing” to get valid probability.
- Training maximizes probability of training data subject to regularization which encourages low or sparse weights.
- Pros:
    - A simple yet low-bias classifier;
    - unlike Naïve Bayes not confounded by diverse, correlated features
- Cons:
    - Slow to train;
    - Some feature scaling issues;
    - Choosing regularisation a nuisance but important since overfitting is a big problem

Support vector machines

- Finds hyperplane which separates the training data with maximum margin.
- Pros:
    - Fast and accurate linear classifier;
    - Can do non-linearity with kernel trick;
    - Works well with huge feature sets
- Cons:
    - Multi-class classification awkward;
    - Feature scaling can be tricky;
    - Deals poorly with class imbalances;
    - Uninterpretable

K-Nearest Neighbour

- Classify based on majority class of k-nearest training examples in feature space (Euclidean distance / Cosine distance)
- Pros:
    - Simple, effective;
    - No training required;
    - Inherently multiclass;
    - Optimal with infinite data
- Cons:
    - Have to select k;
    - Issues with unbalanced classes;
    - Often slow (need to find those k-neighbours);
    - Features must be selected carefully

Decision tree

- Construct a tree where nodes correspond to tests on individual features.
- Pros:
    - In theory, very interpretable;
    - Fast to build and test;
    - Feature representation/scaling irrelevant;
    - Good for small feature sets, handles non-linearly-separable problems
- Cons:
    - In practice, often not that interpretable;
    - Highly redundant sub-trees;
    - Not competitive for large feature sets

Random forests

- An ensemble classifier, Final class decision is majority vote of sub-classifiers
- Pros:
    - Usually more accurate and more robust than decision trees
    - training easily parallelised
- Cons:
    - Same negatives as decision trees
    - too slow with large feature sets

Neural Networks

- An interconnected set of nodes typically arranged in layers.
- Pros:
    - Extremely powerful
    - State-of-the-art accuracy
- Cons:
    - Not an off-the-shelf classifier
    - Very difficult to choose good parameters
    - Slow to train
    - Prone to overfitting

HyperParameter tuning

- Regularization hyperparameters penalize model complexity, used to prevent overfitting.
- For multiple hyperparameters, use grid search.

Evaluation

- Accuracy = correct classifications/total classifications
- Precision = tp / (tp + fp)
- Recall = tp / (tp + fn)
- F1 = 2 precision*recall/(precision + recall)

Lecture 8: N-gram language models

Language models
Assign aprobability to a sequence of words.

Maximum Likelihood estimation
Estimate based on counts in our corpus

For unigram models,

$$P(w_i) = \frac{C(w_i)}{M}$$

For bigram models,

$$P(w_i|w_{i-1}) = \frac{C(w_{i-1}w_i)}{C(w_{i-1})}$$

For n-gram models generally,

$$P(w_i|w_{i-n+1} \dots w_{i-1}) = \frac{C(w_{i-n+1} \dots w_i)}{C(w_{i-n+1} \dots w_{i-1})}$$

Several problems

- Language has long distance effects (need a large n)
- Resulting probabilities are often very small (Use log probability to avoid numerical underflow)
- No probabilities for unseen words (Need to smooth the LM)

Laplacian (Add-one) smoothing

For unigram models (**V**= the vocabulary),

$$P_{add1}(w_i) = \frac{C(w_i) + 1}{M + |\boldsymbol{V}|}$$

For bigram models,

$$P_{add1}(w_i|w_{i-1}) = \frac{C(w_{i-1}w_i) + 1}{C(w_{i-1}) + |\boldsymbol{V}|}$$

Kneser-Ney smoothing
Back-off and Interpolation

$$P_{BO}(w_i|w_{i-2}, w_{i-1}) = \begin{cases} P^*(w_i|w_{i-2}, w_{i-1}) & if\ C(w_{i-2}, w_{i-1}, w_i) > 0 \\ \alpha(w_{i-2}, w_{i-1}) * P_{BO}(w_i|w_{i-1}) & otherwise \end{cases}$$

$$P_{interp}(w_i|w_{i-2}, w_{i-1}) = \lambda(w_{i-2}, w_{i-1})\, P(w_i|w_{i-2}, w_{i-1}) + (1 - \lambda(w_{i-2}, w_{i-1}))P_{interp}(w_i|w_{i-1})$$

Perplexity

$$PP(w_1, w_2, ..\, w_m) = \sqrt[m]{\frac{1}{P(w_1, w_2, ..\, w_m)}}$$

Lecture 9: Lexical Semantics

Lexicalsemantics
How the meanings of words connect to one another.
Manually constructed resources: lexicons, thesauri, ontologies, etc.

Basic Lexical Relations
- Synonyms (same) and antonyms (opposite/complementary)
- Hypernyms (generic), hyponyms (specific)
- Holonyms (whole) and meronyms (part)

Word similarity with paths
simpath(c1,c2) = 1/pathlen(c1,c2)

Wu & Palmer similarity

$$\text{simwup}(c_1,c_2) = \frac{2*\text{depth}(\text{LCS}(c_1,c_2))}{\text{depth}(c_1) + \text{depth}(c_2)}$$

Lin Similarity

* P(c): prob. that word in corpus is instance of concept *c*

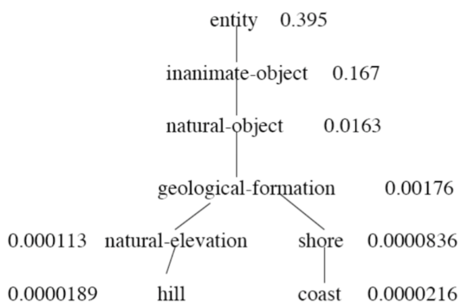
$$P(c) = \frac{\sum_{w \in words(c)} count(w)}{N}$$

* information content (IC)

$$IC(c) = -\log P(c)$$

* Lin distance

$$\text{simlin}(c_1,c_2) = \frac{2*\text{IC}(\text{LCS}(c_1,c_2))}{\text{IC}(c_1) + \text{IC}(c_2)}$$



Word sense disambiguation
- Supervised WSD
  - context is ambiguous
  - How big should context window be?
- Less supervised WSD
  Choose sense whose dictionary gloss from WordNet most overlaps with the context.

Much modern work attempts to derive semantic information directly from corpora, without human intervention.

Lecture 10: Distributional Semantics

Lexical databases

- Manually constructed
    - Expensive
    - Human annotation can be biased and noisy
- Language is dynamic
    - New words: slang, terminology, etc.
    - New senses

Distributional semantics

- Document co-occurrence often indicative of topic (document as context)
- Local context reflects a word's semantic class (word window as context)

Two approaches:

- Count-based (Vector Space Models)
- Prediction-based

Manipulating the VSM

- Weighting the values
- Creating low-dimensional dense vectors
- Comparingvectors

Dimensionality reduction

- Singular value Decomposition (A = U Σ V)
- latent semantic analysis (Truncating)

Words as context

- Lists how often words appear with other words.
- The obvious problem with raw frequency: dominated by common words

Pointwise mutual information

$$PMI(x, y) = \log_2 \frac{p(x, y)}{p(x)p(y)}$$

Skip-gram: Factored Prediction

- Word embeddings should be similar to embeddings of neighbouring words
- Dissimilar to other words that don't occur nearby

Lecture 11: Part of speech tagging

POS Open classes
Nouns / Verbs / Adjectives / Adverbs

POS Closed classes
Prepositions / Determiners / Pronouns / Conjunctions / Modals

Automatic Taggers
- Rule-based taggers
- Statistical taggers
  - Unigram tagger
  - Classifier-based taggers
  - Hidden Markov Model (HMM) taggers

Lecture 12: Neural sequence models

FF-NN for Tagging
5 inputs: 3 x word embeddings and 2 x tag embeddings
1 output: vector of size |T|, using softmax

$$-\sum_i \log P(t_i | w_{i-2}, w_{i-1}, w_i, t_{i-2}, t_{i-1})$$

Recurrent NNLMS
Tagging can be benefit from context to left and right

Pros:
- Robust to word variation, typos, etc
- Excellent generalization, especially RNNs
- Flexible — forms the basis for many other models

Cons:
- Much slower than counts... but GPU acceleration
- Lots of classes (e.g., vocabulary)
- Not good for rare words... but pre-training on big corpora
- Data hungry, not so good on tiny data sets

Lecture 13: Information Extraction

Machine learning in IE

- Named Entity Recognition(NER):
  - sequence models such as seq. classifiers, HMMs or CRFs.
- Relation Extraction:
  - mostly classifiers, either binary or multi-class.

Dealing with adjacent entities: IOB tagging

Relation extraction

- Fixed relation:
  - Rule-based
  - Supervised
  - Semi-supervised
  - Distant supervision
- Open relation:
  - Unsupervised
  - OpenIE

Temporal expressions

Anchoring: Informationusually present in metadata.
Normalisation: mapping expressions to canonical forms.

Event extraction

Event ordering

Lecture 14: Question Answering

Definition:
Question Answering ("QA") is the task of automatically determining the answer (set) for a natural language question.

Question Processing

- Find key parts of question that will help retrieval.
- May reformulate question using templates.
- Predict expected answer type.

Answer Extraction
Find a concise answer to the question, as a span in the text
Framed as classification

QA over structured KB
Natural language querying against knowledge bases using question parsing and logical inference.

Lecture 15: Sequence Tagging: Hidden Markov Models

HMMs for Tagging
Transition Matrix / Emission (observation) Matrix

The Viterbi algorithm

- Complexity:O(T2N),whereTisthesizeofthetagset and N is the length of the sequence.
- Because of the independence assumptions that decompose the problem (specifically, the Markov property).
- Good practice: work with log probabilities to prevent underflow (multiplications become sums).
- HMM is generative models.

Lecture 16: Formal Language Theory & Finite State Automata

What is a "language"?
a set of acceptable strings (e.g., sentences)

Formal Language Theory
Formal apparatus to answer this question automatically, using a grammar.

Key operations

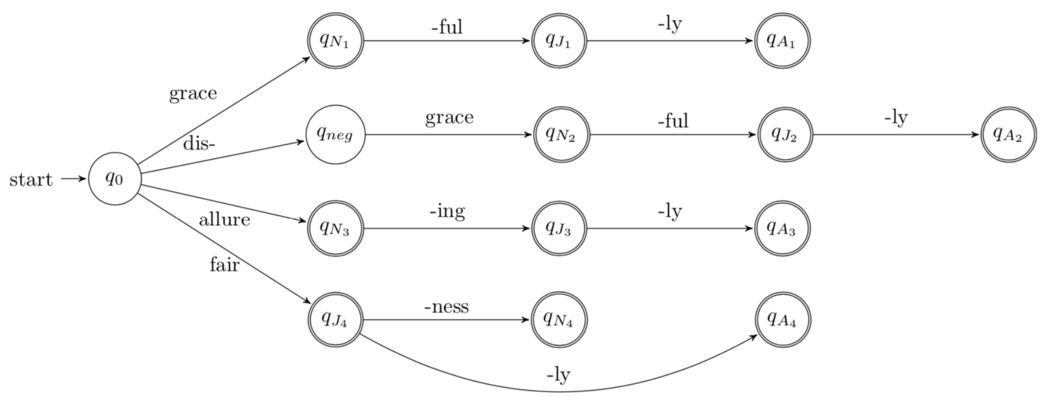
- Membership
  - is the string part of the language?
- Scoring (requires weighting)
  - relax question to graded membership, how good an example of language is the string? (returning a number)
- Transduction
  - input one string, output another
  - A form of translation, but used extensively e.g., tagging = translating from words to tags

Accepted by regular expression which supports the following operations:

- Symbol drawn from alphabet, Σ
- Empty string, ε
- Concatenation of two regular expressions, RS
- Alternation of two regular expressions, R|S
- Kleene star for 0 or more repeats, R*
- Parenthesis () to define scope of operations

Finite State Acceptors
Accepts strings if there is path from q0 to a final state with transitions matching each symbol.

FSA for word morphology



Lecture 17: Context-free Grammars

Basics of Context-free grammars

- Symbols
  - Terminal: word such as book
  - Non-terminal: syntactic label such as NP or NN
  - Convention to use upper and lower-case to distinguish, or else “quotes” for terminals
- Productions (rules) W→XYZ
  - Exactly one non-terminal on left-hand side (LHS)
  - An ordered list of symbols on right-hand side (RHS) can be Terminals or Non-terminals

Regular expressions as CFGs

e.g. [A-Z][a-z]*
S→U S→ULS
U→“A” U→“B” ... U→“Z”
LS→L LS→LLS
L→“a” L→“b” ... L→“z”

The class of regular languages is a subset of the context-free languages, which are specified using a CFG.

CFGs vs regular grammars

- Regular grammars
  - describe a smaller class of languages
  - can be parsed using finite state machines (FSA, FST)
- CFGs
  - can describe hierarchical groupings
  - requires more complex automata to parse (PDA)

CFG trees

- Generation corresponds to a syntactic tree
- Non-terminals are internal nodes
- Terminals are leaves
- Often more than one tree can describe a string

Parsing CFGs

- Bottom-up
  - Start with words, work up towards S
  - CYK parsing
- Top-down
  - Start with S, work down towards words
  - Earley parsing (not covered)

Lecture 18: Probabilistic Parsing

Basics of Probabilistic CFGs

- As for CFGs,same symbol set and same productions.
- In addition, store a probability with each production.
- Probability values denote conditional.
- Each probability must be positive values, between 0 and 1, and the sum must to be 1.
- Grammar / Lexicon

Resolving parse ambiguity

Get the multiplation of all elements in trees.
S in the top-right corner of parse table indicates success

Lecture 19: Dependency Grammar & Parsing

Dependency G vs. Phrase-Structure G

- phrase-structure grammars assume a constituency tree which identifies the phrases in a sentence. Based on idea that these phrases are interchangable (e.g., swap an NP for another NP) and maintain grammaticality.

- Dependency grammar offers a simpler approach: describe binary relations between pairs of words. Namely, between heads and dependents.

What is a Dependency?

- Links between a head word and its dependent words in the sentence: either syntactic roles or modifier relations.

- Dependency tree more directly represents the core of the sentence: who did what to whom?

Dependency tree

- Dependency edges form a tree
    - each node is a word token
    - one node is chosen as the root
    - directed edges link heads and their dependents
- Cf. phrase-structure grammars
    - forms a hierarchical tree
    - word tokens are the leaves
    - internal nodes are 'constituent phrases' e.g., NP, VP etc
- Both use part-of-speech

Projectivity

- A tree is projective if, for all arcs from head to dependent.
- There is a path from head to words that lies between the head and the dependent.
- The tree can be drawn on a plane without any arcs crossing.

Dependency grammar

- In sense of generative grammar.
- Cannot be said to define a language, unlike a context free grammar.
- Any structure is valid, job of probabilistic model to differentiate between poor and good alternatives.
- Many more phrase-structure treebanks, which can be converted into dependencies.

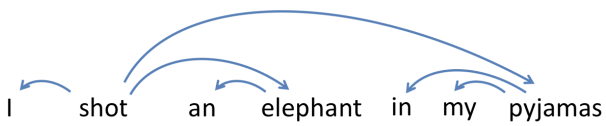
Dependency parsing

- Task of finding the best structure for a given input sentence.
- Graph-based: uses chart over possible parses, and dynamic programming to solve for the maximum.
- Transition-based: treats problem as incremental sequence of decisions over next action in a state machine.

Transition based parsing

- Maintain two data structures
    - buffer = input words yet to be processed
    - stack = head words currently being processed
- Two types of transitions
    - shift = move word from buffer on to top of stack
    - arc = add arc (left/right) between top two items on stack (and remove dependent from stack)
- Always results in a projective tree.

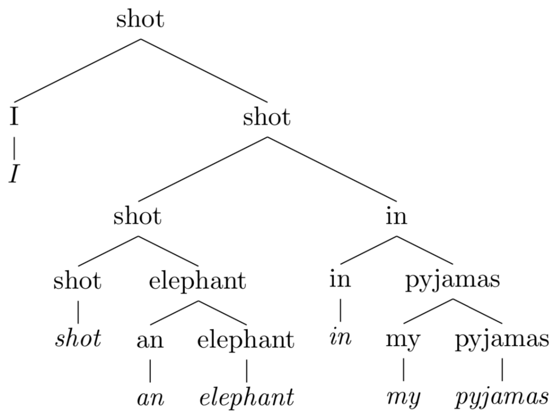
Buffer	Stack	Action
I shot an elephant in my pyjamas		Shift
shot an elephant in my pyjamas	I	Shift
an elephant in my pyjamas	I, shot	Arc-left
an elephant in my pyjamas	shot	Shift
elephant in my pyjamas	shot, an	Shift
in my pyjamas	shot, an, elephant	Arc-left
in my pyjamas	shot, elephant	Arc-right
in my pyjamas	shot	Shift
…	…	…
	shot	<done>

Generated parse: 

- **How do we know when to arc and whether to add left or right facing arcs?** Uses an "oracle" sequence of parser actions. Predict next action in sequence, and update when model disagrees with gold action.

Graph based parsing

- Can consider as a CFG, where lexical items (heads) are non-terminals.
- Score of parse assumed to decompose into pairwise dependencies.
- production shot → shot in means arc-right from "shot" to "in".



Lecture 20: Discourse

Discourse
a coherent, structured group of sentences (utterances)

Discourse segmentation
Assumption: text can be divided into a number of discrete, contiguous sections.
Task: classifying whether a boundary exists between any two sentences.

An unsupervised approach (Text Tiling)
looking for points of low lexical cohesion.

Supervised discourse segmentation
Apply a binary classifier to identify boundaries.
- distributional semantics
- coreference cues
- discourse markers

Discourse parsing
- A proper discourse must be coherent
- Discourse units (DUs) are related by specific coherence relations
- Two related DUs form a new DUs
- All DUs in a coherent discourse must be related
- A discourse will form a tree, which can be parsed

Anaphors
linguistic expressions that refer back to earlier elements in the text

Antecedent Restrictions
- Pronouns must agree in number with their antecedents
- Pronouns must agree in gender with their antecedents
- Pronouns whose antecedents are the subject of the same syntactic clause must be reflexive (...self)

The Centering Algorithm
at any given moment, discourse is focused on a single entity, the “center”.

Lecture 21: Machine translation: word-based models

Noisy channel MT

- eˆ = argmaxe P (e) P(f|e)
- P(f|e) rewards good translations,but permissive of disfluent e
- P(e) rewards e which look like fluent English, and helps put words in the correct order.

How to learn the LM and TM

LM: based on text frequencies in large monolingual corpora (as seen in previous lecture)
TM: based on word co-occurrences in parallel texts

maximum likelihood estimator

Lecture 22: Machine translation: phrase based & Neural Encoder- decoder

Phrase based MT

Treats n-grams as translation units, referred to as ‘phrases’ (not linguistic phrases, just adjacent words)

Finding & scoring phrase pairs

- “Extract”phrasepairsas contiguous chunks in word aligned text;
- Compute counts over the whole corpus;
- Normalise counts to produce ‘probabilities’;

Neural Machine translation

sequence 2 sequence (encoder and decoder)
MT Evaluation: BLEU