

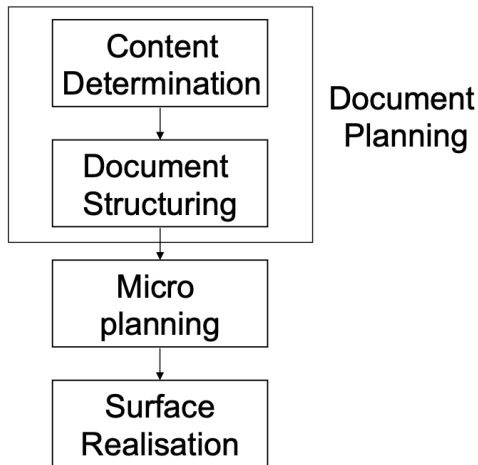
COM6115: Text Processing

Natural Language Generation 3

Chenghua Lin

Department of Computer Science
University of Sheffield

The Architectural View



- Second stage of NLG
 - ◊ Choosing language to express content
- Several subtasks
 - ◊ Lexical choice: Which words to use
 - ◊ Reference: How to refer to objects
 - ◊ Aggregation: How/when combine phrases into sentences

- Problem: There are zillions of ways of expressing a message in words
 - ◇ John sold the book to Mary
 - ◇ Mary bought the book from John
 - ◇ John sold the book. Mary bought it
 - ◇ Etc, etc
- Which one should we use?

- Theoretical
 - ◇ Define what “best” means, make microplanning choices that optimise it
 - ◇ Hard to do in practice because we don't have good models of the effects of choices
- Pragmatic
 - ◇ Imitate corpus
 - Use statistical learning if corpus large enough
 - ◇ Problem: sometimes corpus texts may not be very good from a microplanning perspective

- Lexical choice: the task of choosing the right words or lemmas to express the contents of the message
- I.e., which word should be used to communicate a concept?
 - ◇ Buy vs sell
 - ◇ Ascended vs rose vs surfaced
 - ◇ Too fast vs too rapidly
 - ◇ Recommend vs suggest
 - ◇ etc


Issues that affect lexical choice

- Frequency (affects readability)
 - ◇ lie vs prevarication
- Formality:
 - ◇ Error vs howler
- Focus, expectations
 - ◇ not many, few, a few, only a few [students failed the exam]
- Technical terms
 - ◇ (statistics) standard error, not
 - ◇ standard mistake
- Convention
 - ◇ Temperature falls, Wind speed eases

Statistics-Based Lexical Choice for NLG from Quantitative Information

- NLG systems express information in human language

| Forecasted numeric data | | | | |
|--------------------------|--------------------|--------------|--|--|
| Wind Direction (azimuth) | Wind Speed (knots) | Gust (knots) | | |
| 2 | 9 | 11 | | |
| 92 | 20 | 30 | | |
| 130 | 4 | 5 | | |



| Forecast Text |
|------------------|
| SE 10-12 |
| E 20-22 GUSTS 30 |
| MAINLY 8 OR LESS |

- Systems need to “know” what expressions are most suitable for expressing a given piece of information.

wd=130



“SE”

wd=92



“E”

ws=4

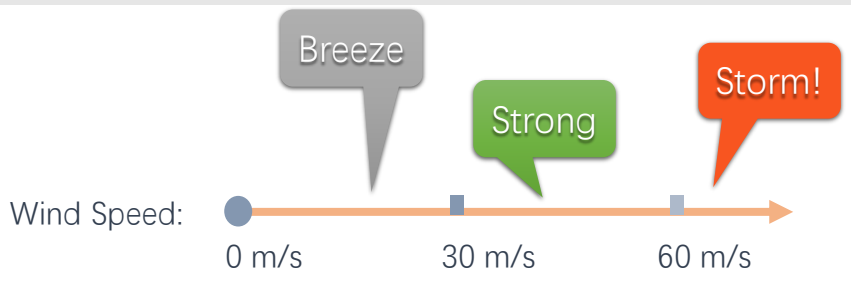


“8 OR LESS”



Motivation

- Systems need to “know” what expressions are most suitable for expressing a given piece of information.



- To develop a statistical algorithm for lexical choice for quantitative information, which can
 - ◇ Detect the relationship between data dimensions (aka. attributes) and words
 - ◇ Does not rely on hand-crafted rules;
 - ◇ Predict both when and which word(s) should be used;
 - ◇ One word can refer to multiple dimensions.

$P(\text{"muggy"} \mid \text{ws}=20, \text{temp}=35, \text{humid}=97, \dots)$

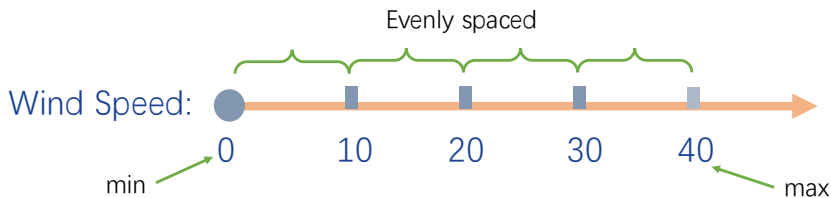
- Each data record consists of attribute-value pairs.
- E.g., dir=2,ws=9,gusts=11, where the attributes are “dir”, “ws”, and “gusts”.

| Forecasted | numerical | data |
|--------------------------------|--------------------------|---------------------|
| Wind Direction (azimuth) | Wind Speed (knots) | Gust (knots) |
| 2 | 9 | 11 |
| 92 | 20 | 30 |
| 130 | 4 | 5 |

Representing Data in Vector

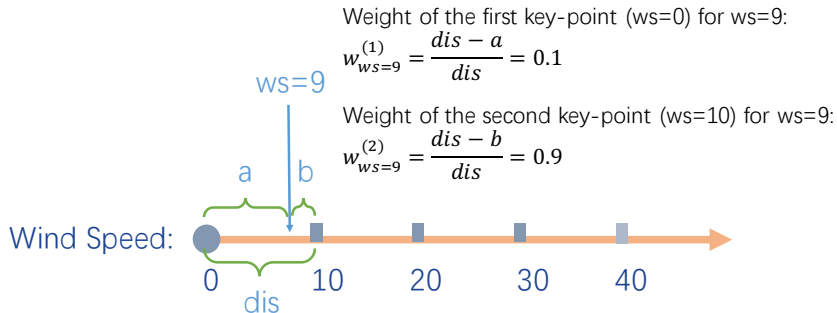
We represent each attribute (e.g. wind speed) as a combination of some weighted **key-points**.

- The key-points are derived by:
 - ◇ Taking the min and max values of the attribute (from training data)
 - ◇ Key-points are evenly spaced between the min and max values
- The number of the key-points for an attribute are fixed



Data Representation

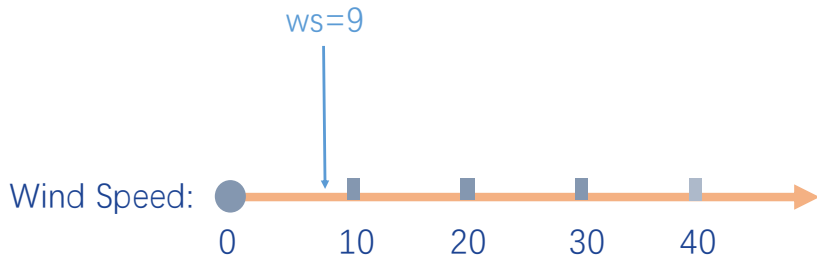
An example of deriving the key point weights for the attribute value $ws=9$ (i.e., wind speed dimension).



Data Representation

An example of deriving the key point weights for the data record $ws=9$ (i.e., wind speed dimension).

- In this way, an attribute value (e.g. $ws=9$) can be represented by a key point weight vector
- I.e. the weight vector of $ws=9$ is $[0.1, 0.9, 0, 0, 0]$.



Representing Data in Vector

Similarly, a data record (i.e., a set of attribute-value pairs) can be represented by multiple groups of key-points, e.g.:

$$ws = 9 \rightarrow [0.1, 0.9, 0, 0, 0]$$

$$dir = 2 \rightarrow [0.97, 0.03, 0, 0, 0]$$

Thus, to represent a set of attribute-value pairs, we concatenate the individual weight vectors, e.g.:

$$\{ws = 9, dir = 2\} \rightarrow [0.1, 0.9, 0, 0, 0, 0.97, 0.03, 0, 0, 0]$$

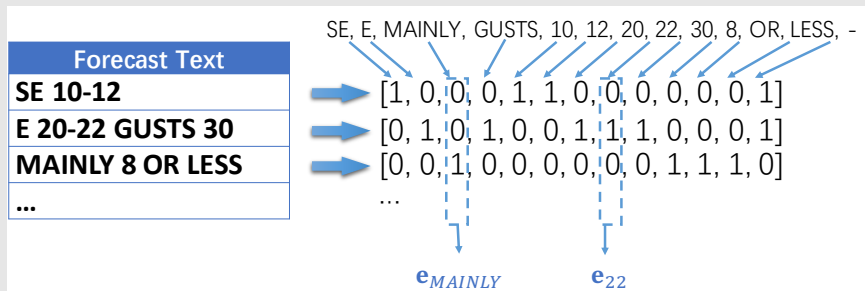
Representing Data in Vector

The entire data-text corpus can then be represented with a vector matrix (\mathbf{K}), whose row corresponds to the weight vector of a data record.

$$\begin{array}{l} \{ws = 9, dir = 2, \dots\} \\ \{ws = 20, dir = 130, \dots\} \\ \{ws = 2, dir = 90, \dots\} \\ \dots \end{array} \rightarrow \mathbf{K} = \begin{array}{c} \begin{array}{cc} \text{Wind speed} & \text{Wind direction} \end{array} \\ \begin{array}{ccccccccc} \hline & \underbrace{\hspace{2cm}} & & \underbrace{\hspace{2cm}} & & \\ \hline \end{array} \begin{bmatrix} 0.1 & 0.9 & 0 & 0 & 0 & 0.97 & 0.03 & 0 & 0 & 0 & \dots \\ 0 & 0 & 1 & 0 & 0 & 0 & 0.66 & 0.44 & 0 & 0 & \dots \\ 0.8 & 0.2 & 0 & 0 & 0 & 1 & 0.86 & 0.14 & 0 & 0 & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \end{bmatrix} \end{array}$$

Representing Text

- We use a column vector (namely \mathbf{e}_i) to represent the text of a data record. Each element of \mathbf{e}_i indicates whether a word appears in the data record, e.g.:



Representing Words in Vector

- So far, data are represented by weight vectors, whose values can be calculated using key points.
- We represent words in the corpus using the same weight vectors whose values are unknown.

$$\{ws = 9, dir = 2\} \rightarrow v_{data} = [0.1, 0.9, 0, 0, 0, 0.97, 0.03, 0, 0, 0]$$
$$"muggy" \rightarrow v_{muggy} = [?, ?, ?, ?, ?, ?, ?, ?, ?, ?]$$

Task: To estimate v_i for word i given a data-to-text corpus as input.

Assumption: v_i and v_d should be close to each other in the vector space if word i appears in data record d

$$\frac{v_{d1} \cdot v_i}{\|v_{d1}\| \|v_i\|} = \text{appear}(i, d1)$$

$$\frac{v_{d2} \cdot v_i}{\|v_{d2}\| \|v_i\|} = \text{appear}(i, d2)$$

...

NB: $\text{appear}(i, d1) = 1$ if word i appears in data record d and 0 otherwise.

Our task is to find the weight vector v_i for each word i , such that the similarity of v_i and v_d is close to $appear(i, d)$ as much as possible for each data record (d).

$$v_i = \min_{v_i} \sqrt{\sum_d (\text{sim}(v_i, v_d) - \text{appear}(i, d))^2}$$

- Finding v_i equivalent to finding the optimal solution the following equation using least squares

$$\mathbf{K}' \cdot \frac{v_i}{\|v_i\|} = \mathbf{e}_i$$
$$\text{opt}\left(\frac{v_i}{\|v_i\|}\right) = (\mathbf{K}'^T \mathbf{K}')^{-1} \mathbf{K}'^T \mathbf{e}_i$$

- Once v_i is solved, we can then estimate the most appropriate words for for unseen data .

| Input data | |
|----------------------|-------|
| Month | Oct |
| Cloud Cover | 11% |
| Precipitation | 0 mm |
| Temperature | 25° C |



| Output | |
|-------------------|--------|
| Word | Weight |
| sunny | 3.908 |
| bright | 3.797 |
| warm | 2.852 |
| lovely | 2.61 |
| moody | 2.477 |
| hot | 2.3 |
| dependable | 2.093 |
| muggy | 1.978 |
| calm | 1.933 |
| clear | 1.804 |
| autumny | 1.699 |

- Which phrase should be used to identify an object?
- Referring expression generation: the task of selecting the content (and, to some extent, the form) of referential noun phrases in text.
 - ◊ Look at the big dog
 - ◊ Look at Fido
 - ◊ Look at it

Types of reference

- Pronoun – it, them, him, you,...
- Name – Dr Adam Smith, Adam Smith, Adam, Dr Smith
- Definite NP – the big black dog, the big dog, the black dog, the dog

- Use pronoun if possible
 - ◇ Referent mentioned recently
 - ◇ Pronoun is not ambiguous
- Else use name if possible
 - ◇ Shortest form which is unambiguous and stylistically allowed
- Else use definite NP
 - ◇ Shortest one, prefer basic-level words
- Only use forms seen in corpus

- Aggregation: the task of merging distinct representations into a single, more concise representation
- When/how should we combine phrases?
 - ◇ Your first ascent was fine. Your second ascent was fine.
 - ◇ Your first ascent was fine, and your second ascent was fine.
 - ◇ Your first ascent and your second ascent were fine.
 - ◇ Your first and second ascents were fine.

Suggestions on Aggregation

- Generally use the deepest one we can
 - ◇ Your first ascent was safe, and your second ascent was safe.
 - ◇ Your first ascent and your second ascent were safe.
 - ◇ Your first and second ascents were safe.
- Depends on how similar phrases are.
- Depends on genre (corpus)

- Decide how to best express a message in language
 - ◊ Essential for producing “nice” texts
- Imitating corpus works to some degree, but not perfectly
 - ◊ Currently more of an art than a science
- Key is better understanding of how linguistic choices affected readers
 - ◊ Our SumTime weather-forecast generator microplans better than human forecasters

- Third (last) NLG stage
- Creating linear text from (typically) structured input; ensuring syntactic correctness
- Take care of details of language
 - ◇ Syntactic details
 - Eg Agreement (the dog runs vs the dogs run)
 - ◇ Morphological details
 - Eg, plurals (dog/dogs vs box/boxes)
 - ◇ Presentation details
 - Eg, fit to 80 column width

- Problem: There are lots of finicky details of language which most people developing NLG systems don't want to worry about
- Solution: Automate this using a realiser

- Sentences must obey the rules of English grammar
 - ◇ Specifies which order words should appear in, extra function words, word forms
- Many aspects of grammar are somewhat bizarre
- Just tell realiser verb, tense, whether negated, and it will figure out the verb group
 - ◇ (watch, future) -> will watch
 - ◇ (watch, past, negated) -> did not watch
 - ◇ Etc
- Similarly automate other “obscure” encodings of information

In linguistics, morphology is the study of words, how they are formed, and their relationship to other words in the same language. E.g.,

- Variations of a root form of a word, e.g., prefixes, suffixes
- Inflectional morphology - same core meaning
 - ◇ plurals, past tense, superlatives, e.g., dog, dogs
 - ◇ part of speech unchanged
- Derivational morphology - change meaning
 - ◇ prefix *re* means do again: reheat, resit
 - ◇ suffix *er* means one who: teacher, baker
 - ◇ part of speech changed

- Calculates morphological variants automatically
 - ◇ (dog, plural) -> dogs
 - ◇ (box, plural) -> boxes
 - ◇ (child, plural) -> children
 - ◇ etc
- Automatically insert appropriate punctuation for a structure
- Many possible output formats
 - ◇ Simple text
 - ◇ HTML
 - ◇ MS Word

- simpleNLG – relatively limited functionality, but well documented, fast, easy to use, tested
 - ◊ Most popular, easy-to-use, programmatically controllable and extendable realisation engine.
 - ◊ Has adapted into many (western) languages: French, German, Mandarin ...
- KPML – lots of functionality but poorly documented, buggy, slow
- openCCG – somewhere in between
- Many more

Realiser

- creates linear text from (typically) structured input; ensuring syntactic correctness
- automates the finicky details of language
 - ◇ So NLG developer doesn't have to worry about these
 - ◇ One of the advantages of NLG