Language and Computers

Writers' Aids

L245
(Based on Dickinson, Brew, & Meurers (2013))

Spring 2017

Spelling & grammar correction

We are all familiar with spelling & grammar correctors

- They are used to **improve document quality**
- They are not typically used to **provide feedback**

Typically designed for native speakers of a language

- Next unit (Language Tutoring Systems): feedback for non-native speakers

Why people care about spelling

- Misspellings can cause misunderstandings
- Standard spelling makes it easy to organize words & text:
  - e.g., Without standard spelling, how would you look up things in a lexicon or thesaurus?
  - e.g., Optical character recognition software (OCR) can use knowledge about standard spelling to recognize scanned words even for hardly legible input
- Standard spelling makes it possible to provide a single text, accessible to a wide range of readers (different backgrounds, speaking different dialects, etc.)
- Using standard spelling can make a good impression in social interaction

Use of writers' aids

How are spell checkers (and grammar checkers) used?

- **Interactive spelling checker**: spell checker detects errors as you type
  - It may or may not make suggestions for correction
  - It needs a "real-time" response (i.e., must be fast)
  - It is up to the human to decide if the spell checker is right or wrong, and so we may not require 100% accuracy (especially with a list of choices)
- **Automatic spelling corrector**: spell checker runs on a whole document, finds errors, and corrects them
  - A more difficult task
  - A human may or may not proofread the results later

Outline

Tasks are typically divided into:

- **Error detection** = simply find the misspelled words
- **Error correction** = correct the misspelled words
  - e.g., *ater* is a misspelled word, but what is the correct word? *water*? *later*? *after*?

We will consider three types of techniques:

- Non-word error detection
- Isolated-word error detection & correction
- Grammar correction (Context-dependent word error detection & correction)

Non-word error detection

- **Word recognition**: split up "words" into true words and non-words
  - **Non-word error detection**: detect the non-words
- How is non-word error detection done?
  - Using a dictionary (construction and lookup)
  - *n*-gram analysis (more for OCR error detection)

Dictionaries

Intuition:

- Have a complete list of words and check the input words against this list.
- If it's not in the dictionary, it's not a word.

Two aspects:

- **Dictionary construction**: build the dictionary (what do you put in it?)
- **Dictionary lookup**: look up a potential word in the dictionary (how do you do this quickly?)

Dictionary construction

One set of issues: who is the dictionary for?

- **Domain-specificity**: only contain words relevant to the user
- **Dialectal consistency**: only include forms for one variety of a language (e.g., American *color* or British *colour*)

Another set of issues: how do we analyze words?

- **Tokenization**: What is a word?
- **Inflection**: How are some words related?
- **Productivity of language**: How many words are there?

Addressing these issues determines how to build dictionary

Challenges for spelling correction
Tokenization

Tokenization splits a sentence into its component words

Intuitively, a "word" is simply whatever is between two spaces, but this is not always so clear.

- Contractions: two words combined into one
  - e.g., *can't*, *he's*, *John's [car]* (vs. *his car*)
- Multi-word expressions: single term with space(s)
  - e.g., *New York*, *in spite of*, *déjà vu*
- Hyphens (ambiguous if a hyphen ends a line)
  - Some are always a single word: *e-mail*, *co-operate*
  - Others are two words combined into one: *Columbus-based*, *sound-change*
- Abbreviations: may stand for multiple words
  - e.g., *etc.* = *et cetera*, *ATM* = *Automated Teller Machine*

Challenges for spelling correction
Inflection

- A word in English may appear in various guises due to word **inflections** = word endings which are fairly systematic for a given part of speech
  - Plural noun ending: *the boy* + *s* → *the boys*
  - Past tense verb ending: *walk* + *ed* → *walked*
- Challenges for spell checking:
  - Exceptions to the rules: **mans*, **runned*
  - Words which look like they have a given ending, but they don't: *Hans*, *deed*

Challenges for spelling correction
Productivity

Productivity means that language allows for new words

- Words entering and exiting the lexicon, e.g.:
  - *thou*, or *spleet* 'split' (*Hamlet III.2.10*) moving out
  - New words all the time: *jeggings*, *drumble*, *retweet*, ...
- Part of speech change: nouns ↔ verbs
  - *retweeting* can be formed off the noun *retweet*
- Morphological productivity: addition of prefixes & suffixes
  - e.g., I can speak of *un-email-able* for someone who you can't reach by email.

N-gram analysis

Idea: use typical phonotactic patterns to identify words

- An **n-gram** here is a string of *n* letters.

a	1-gram (unigram)
at	2-gram (bigram)
ate	3-gram (trigram)
late	4-gram
⋮	⋮

- We can use this n-gram information to define what the possible strings in a language are.
  - e.g., *po* is a possible English string, whereas *kvt* is not.

This is more useful to correct optical character recognition (OCR) output, but we'll still take a look.

Bigram array

- **Bigram array**: bigram information stored in a table
- An example, for the letters *k*, *l*, *m*, with examples in parentheses

	…	k	l	m	…
⋮					
k		0	1 (*tac**kl**e*)	1 (*Hac**km**an*)	
l		1 (*e**lk***)	1 (*he**ll**o*)	1 (*a**lm**s*)	
m		0	1 (*ha**ml**et*)	1 (*ha**mm**er*)	
⋮					

- The first letter of the bigram is given by the vertical letters (i.e., down the side), the second by the horizontal
- This is a **non-positional bigram array**: the array 1s and 0s apply for a string found anywhere within a word (beginning, 4th character, ending, etc.).

Language and Computers
Writers' Aids
Introduction
Non-word error detection
Dictionaries
N-gram analysis
Isolated-word error correction
Types of errors
Rule-based methods
Similarity key techniques
Minimum edit distance
Probabilistic methods
Error correction for web queries
Grammar correction
Syntax and Computing
Grammar correction rules
Caveat emptor

Positional bigram array

- To store information specific to the beginning, the end, or some other position in a word, use a **positional bigram array**: the array only applies for a given position in a word.
- Here's the same array as before, but now only applied to word endings:

	…	k	l	m	…
⋮					
k		0	0	0	
l		1 (*e**lk***)	1 (*ha**ll***)	1 (*e**lm***)	
m		0	0	0	
⋮					

Language and Computers
Writers' Aids
Introduction
Non-word error detection
Dictionaries
N-gram analysis
Isolated-word error correction
Types of errors
Rule-based methods
Similarity key techniques
Minimum edit distance
Probabilistic methods
Error correction for web queries
Grammar correction
Syntax and Computing
Grammar correction rules
Caveat emptor

Isolated-word error correction

- Having discussed how errors can be detected, we want to know how to correct these misspelled words:
  - **Isolated-word error correction**: correcting words without taking context into account
  - This technique can only handle errors resulting in non-words
- Knowledge about what is a typical error helps in finding correct word
  - What leads to errors? What properties do errors have?

Language and Computers
Writers' Aids
Introduction
Non-word error detection
Dictionaries
N-gram analysis
Isolated-word error correction
Types of errors
Rule-based methods
Similarity key techniques
Minimum edit distance
Probabilistic methods
Error correction for web queries
Grammar correction
Syntax and Computing
Grammar correction rules
Caveat emptor

Types of errors

Keyboard effects

Keyboard proximity

- e.g., *program* might become *progrsm* since *a* and *s* are next to each other on a QWERTY keyboard

Space bar issues

- **Run-on** errors: two separate words become one
  - e.g., *the fuzz* becomes *thefuzz*
- **Split** errors: one word becomes two separate items
  - e.g., *equalization* becomes *equali zation*
  - The resulting items might still be words: e.g., *a tollway* becomes *atoll way*

Language and Computers
Writers' Aids
Introduction
Non-word error detection
Dictionaries
N-gram analysis
Isolated-word error correction
Types of errors
Rule-based methods
Similarity key techniques
Minimum edit distance
Probabilistic methods
Error correction for web queries
Grammar correction
Syntax and Computing
Grammar correction rules
Caveat emptor

Types of errors

Phonetic errors

Phonetic errors

Errors stemming from imperfect sound-letter correspondences

- **Homophones**: two words which sound the same
  - e.g., *red*/*read* (past tense), *cite*/*site*/*sight*, *they're*/*their*/*there*
- Substitutions: replacing a letter (or sequence) with similar-sounding one
  - e.g., *seperate* (for *separate*)

Language and Computers
Writers' Aids
Introduction
Non-word error detection
Dictionaries
N-gram analysis
Isolated-word error correction
Types of errors
Rule-based methods
Similarity key techniques
Minimum edit distance
Probabilistic methods
Error correction for web queries
Grammar correction
Syntax and Computing
Grammar correction rules
Caveat emptor

Types of errors

Knowledge-based errors

- Not knowing a word:
  - e.g., *boocoo* (for *beaucoup*),
- Not knowing a rule:
  - e.g., consonant (non-)doubling: *labeled* vs. *labelled*, *hoped* vs. *hopped*
- Knowing something is odd about the spelling, but guessing the wrong thing
  - e.g., *siscors* (for *scissors*)

Describing typical errors

Errors can be examined under a more mechanistic lens:

Types of operations

- **insertion** = a letter is added to a word
- **deletion** = a letter is deleted from a word
- **substitution** = a letter is put in place of another one
- **transposition** = two adjacent letters are switched

Note that the first two alter the length of the word, whereas the second two maintain the same length.

Language and Computers
Writers' Aids
Introduction
Non-word error detection
Dictionaries
N-gram analysis
Isolated-word error correction
Types of errors
Rule-based methods
Similarity key techniques
Minimum edit distance
Probabilistic methods
Error correction for web queries
Grammar correction
Syntax and Computing
Grammar correction rules
Caveat emptor

Typical error properties

- Word length effects: most misspellings are within two characters in length of original
  - When searching for the correct spelling, we do not usually need to look at words with greater length differences
- First-position error effects: the first letter of a word is rarely erroneous
  - When searching for the correct spelling, the process is sped up by being able to look only at words with the same first letter

Language and Computers
Writers' Aids
Introduction
Non-word error detection
Dictionaries
N-gram analysis
Isolated-word error correction
Types of errors
Rule-based methods
Similarity key techniques
Minimum edit distance
Probabilistic methods
Error correction for web queries
Grammar correction
Syntax and Computing
Grammar correction rules
Caveat emptor

Isolated-word error correction methods

- Many different methods are used; we will briefly look at four methods:
  - Rule-based methods
  - Similarity key techniques
  - Probabilistic methods
  - Minimum edit distance
- The methods play a role in one of the three basic steps:
  1. Detection of an error (discussed above)
  2. Generation of candidate corrections
     - rule-based methods
     - similarity key techniques
  3. Ranking of candidate corrections
     - probabilistic methods
     - minimum edit distance (also usable for generation)

Language and Computers
Writers' Aids
Introduction
Non-word error detection
Dictionaries
N-gram analysis
Isolated-word error correction
Types of errors
Rule-based methods
Similarity key techniques
Minimum edit distance
Probabilistic methods
Error correction for web queries
Grammar correction
Syntax and Computing
Grammar correction rules
Caveat emptor

Rule-based methods

One can generate correct spellings by writing rules:

- Common misspelling rewritten as correct word:
  - e.g., *hte* → *the*
- Rules
  - based on inflections:
    - e.g., *VCing* → *VCCing*, where

      V = letter representing vowel, basically the regular expression `[aeiou]`

      C = letter representing consonant, basically `[bcdfghjklmnpqrstvwxyz]`
  - based on other common spelling errors (such as keyboard effects or common transpositions):
    - e.g., *CsC* → *CaC*
    - e.g., *cie* → *cei*

Language and Computers
Writers' Aids
Introduction
Non-word error detection
Dictionaries
N-gram analysis
Isolated-word error correction
Types of errors
Rule-based methods
Similarity key techniques
Minimum edit distance
Probabilistic methods
Error correction for web queries
Grammar correction
Syntax and Computing
Grammar correction rules
Caveat emptor

Similarity key techniques (SOUNDEX)

- Problem: How can we find a list of possible corrections?
- Solution: Store words in different boxes in a way that puts the similar words together.
- Example:
  1. Start by storing words by their first letter (first letter effect),
     - e.g., *punc* starts with the code P.
  2. Then assign numbers to each letter
     - e.g., 0 for vowels, 1 for *b, p, f, v* (all bilabials), and so forth, e.g., *punc* → P052
  3. Then throw out all zeros and repeated letters,
     - e.g., P052 → P52.
  4. Look for real words within the same box,
     - e.g., *punk* is also in the P52 box.

http://en.wikipedia.org/wiki/Soundex

Language and Computers
Writers' Aids
Introduction
Non-word error detection
Dictionaries
N-gram analysis
Isolated-word error correction
Types of errors
Rule-based methods
Similarity key techniques
Minimum edit distance
Probabilistic methods
Error correction for web queries
Grammar correction
Syntax and Computing
Grammar correction rules
Caveat emptor

Minimum edit distance

- In order to rank possible spelling corrections, it can be useful to calculate the **minimum edit distance** = minimum number of operations it would take to convert one word into another.
- For example, we can take the following five steps to convert *junk* to *haiku*:
  1. *ju**n**k* → *juk* (deletion)
  2. ***j**uk* → ***h**uk* (substitution)
  3. ***hu**k* → *h**k**u* (transposition)
  4. *hku* → *h**i**ku* (insertion)
  5. *hiku* → *h**a**iku* (insertion)
- But is this the minimal number of steps needed?

Language and Computers
Writers' Aids
Introduction
Non-word error detection
Dictionaries
N-gram analysis
Isolated-word error correction
Types of errors
Rule-based methods
Similarity key techniques
Minimum edit distance
Probabilistic methods
Error correction for web queries
Grammar correction
Syntax and Computing
Grammar correction rules
Caveat emptor

Computing edit distances

Figuring out the upper bound

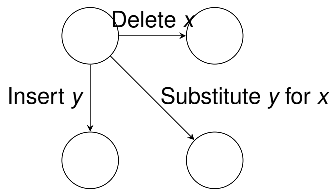
- To be able to compute the edit distance of two words at all, we need to ensure there is a finite number of steps.
- This can be accomplished by
  - requiring that letters cannot be changed back and forth a potentially infinite number of times, i.e., we
  - limit the number of changes to the size of the material we are presented with, the two words.
- Idea: Never deal with a character in either word more than once.
- Result:
  - We could delete each character in the first word and then insert each character of the second word.
  - Thus, we will never have a distance greater than $length(word1) + length(word2)$

Language and Computers
Writers' Aids
Introduction
Non-word error detection
Dictionaries
N-gram analysis
Isolated-word error correction
Types of errors
Rule-based methods
Similarity key techniques
Minimum edit distance
Probabilistic methods
Error correction for web queries
Grammar correction
Syntax and Computing
Grammar correction rules
Caveat emptor

Computing edit distances

Using a graph to map out the options

- To calculate minimum edit distance, we set up a **directed, acyclic graph**, a set of nodes (circles) and arcs (arrows).
- Horizontal arcs correspond to deletions, vertical arcs correspond to insertions, and diagonal arcs correspond to substitutions (a letter can be "substituted" for itself).

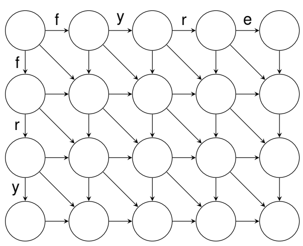


Discussion here based on Roger Mitton's book *English Spelling and the Computer*.

Computing edit distances

An example graph

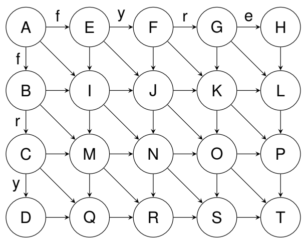
- Say, the user types in *fyre*.
- We want to calculate how far away *fry* is (one of the possible corrections). In other words, we want to calculate the minimum edit distance (or minimum edit cost) from *fyre* to *fry*.
- As the first step, we draw the following directed graph:



Computing edit distances

Adding numbers to the example graph

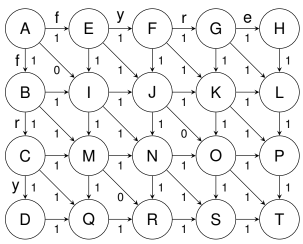
- The graph is **acyclic** = for any given node, it is impossible to return to that node by following the arcs.
- We can add identifiers to the states, which allows us to define a **topological order**
  - Topological order: not every pair of nodes has an ordering



Computing edit distances

Adding costs to the arcs of the example graph

- We need to add the costs involved to the arcs.
- In the simplest case, the cost of deletion, insertion, and substitution is 1 each (and substitution with the same character is free).



- Instead of assuming the same cost for all operations, in reality one will use different costs, e.g., for the first character or based on the confusion probability.

Computing edit distances

How to compute the path with the least cost

We want to find the path from the start (A) to the end (T) with the least cost.

- The simple but dumb way of doing it:
  - Follow every path from start (A) to finish (T) and see how many changes we have to make.
  - But this is very inefficient! There are many different paths to check.

Computing edit distances

The smart way to compute the least cost

- The smart way to compute the least cost uses **dynamic programming**: process designed to make use of results computed earlier
  - We follow the topological ordering & calculate the least cost for each node:
    - We add the cost of an arc to the cost of reaching the node this arc originates from.
    - We take the minimum of the costs calculated for all arcs pointing to a node and store it for that node.
  - The key point is that we are storing partial results along the way, instead of recalculating everything, every time we compute a new path.

Probabilistic methods

When converting from one word to another, a lot of words will be the same distance.

e.g., for the misspelling *wil*, all of the following are one edit distance away:

- *will*
- *wild*
- *wilt*
- *nil*

Probabilities will help to tell them apart

Language and Computers
Writers' Aids
Introduction
Non-word error detection
Dictionaries
N-gram analysis
Isolated-word error correction
Types of errors
Rule-based methods
Similarity key techniques
Minimum edit distance
Probabilistic methods
Error correction for web queries
Grammar correction
Syntax and Computing
Grammar correction rules
Caveat emptor

The Noisy Channel Model

Probabilities can be modeled with the **noisy channel model**

Hypothesized Language: X
$\Downarrow$
Noisy Channel: X $\rightarrow$ Y
$\Downarrow$
Actual Language: Y

Goal: Recover X from Y

- The noisy channel model has been very popular in speech recognition, among other fields

(Thanks to Mike White for the slides on the Noisy Channel Model)

Language and Computers
Writers' Aids
Introduction
Non-word error detection
Dictionaries
N-gram analysis
Isolated-word error correction
Types of errors
Rule-based methods
Similarity key techniques
Minimum edit distance
Probabilistic methods
Error correction for web queries
Grammar correction
Syntax and Computing
Grammar correction rules
Caveat emptor

Noisy Channel Spelling Correction

Correct Spelling: X
$\Downarrow$
Typos, Mistakes: X $\rightarrow$ Y
$\Downarrow$
Misspelling: Y

Goal: Recover correct spelling X from misspelling Y

- Noisy word: Y = observation (incorrect spelling)
- We want to find the word (X) which maximizes: $P(X|Y)$, i.e., the probability of X, given that Y has been seen

Language and Computers
Writers' Aids
Introduction
Non-word error detection
Dictionaries
N-gram analysis
Isolated-word error correction
Types of errors
Rule-based methods
Similarity key techniques
Minimum edit distance
Probabilistic methods
Error correction for web queries
Grammar correction
Syntax and Computing
Grammar correction rules
Caveat emptor

Example

Correct Spelling: *swam*
$\Downarrow$
Transposition: *ld* $\rightarrow$ *dl*
$\Downarrow$
Misspelling: *sawm*

Goal: Recover correct spelling *swam* from misspelling *sawm* (i.e., $P(swam|sawm)$)

Language and Computers
Writers' Aids
Introduction
Non-word error detection
Dictionaries
N-gram analysis
Isolated-word error correction
Types of errors
Rule-based methods
Similarity key techniques
Minimum edit distance
Probabilistic methods
Error correction for web queries
Grammar correction
Syntax and Computing
Grammar correction rules
Caveat emptor

Conditional probability

(Reminder)

$p(x|y)$ is the probability of x given y

- Let's say that it rains appears 20 times in a span of 40 days
  - $p(rain) = 20/40 = 0.5$
- Now, let's say I bring an umbrella to work on 18 (of the 40) days, and it rains on 2 of those days
  - $p(rain|umbrella) = 2/18 = 0.1111$

Note: there is no causation implied here; we are simply counting things

Language and Computers
Writers' Aids
Introduction
Non-word error detection
Dictionaries
N-gram analysis
Isolated-word error correction
Types of errors
Rule-based methods
Similarity key techniques
Minimum edit distance
Probabilistic methods
Error correction for web queries
Grammar correction
Syntax and Computing
Grammar correction rules
Caveat emptor

Bayes Rule

With X as the correct word and Y as the misspelling ...

$P(X|Y)$ is impossible to calculate directly, so we use:

- $P(Y|X)$ = the probability of the observed misspelling given the correct word
- $P(X)$ = the probability of the (correct) word occurring anywhere in the text

Bayes Rule allows us to calculate $p(X|Y)$ in terms of $p(Y|X)$:

(1) **Bayes Rule:** $P(X|Y) = \frac{P(Y|X)P(X)}{P(Y)}$

The Noisy Channel and Bayes Rule

We can directly relate Bayes Rule to the Noisy Channel:

$$\overbrace{Pr(X|Y)}^{Posterior} = \frac{\overbrace{Pr(Y|X)}^{Noisy\ Channel}\ \overbrace{Pr(X)}^{Prior}}{\underbrace{Pr(Y)}_{Normalization}}$$

Goal: for a given y, find x =

$$\arg\max_x \overbrace{Pr(y|x)}^{Noisy\ Channel}\ \overbrace{Pr(x)}^{Prior}$$

The denominator is ignored because it's the same for all possible corrections, i.e., the observed word (y) doesn't change

Finding the Correct Spelling

Goal: for a given misspelling y, find correct spelling x =

$$\arg\max_x \overbrace{Pr(y|x)}^{Error\ Model}\ \overbrace{Pr(x)}^{Language\ Model}$$

1. List "all" possible candidate corrections, i.e., all words with one insertion, deletion, substitution, or transposition
2. Rank them by their probabilities

Example: calculate for *swam*

$$Pr(sawm|swam)Pr(swam)$$

and see if this value is higher than for any other possible correction.

Obtaining probabilities

How do we get these probabilities?

We can count up the number of occurrences of X to get $P(X)$, but where do we get $P(Y|X)$?

- We can use confusion matrices: one matrix each for insertion, deletion, substituion, and transposition

Obtaining probabilities
Confusion probabilities

- It is impossible to fully investigate all possible error causes and how they interact, but we can learn from watching how often people make errors and where.
- One way is to build a **confusion matrix** = a table indicating how often one letter is mistyped for another

		correct				
		...	r	s	t	...
	⋮					
	r		n/a	12	22	
typed	s		14	n/a	15	
	t		11	37	n/a	
	⋮					

(cf. Kernighan et al 1999)

Obtaining probabilities

Using a spelling error-annotated corpus:

- These matrices are calculated by counting how often, e.g., *ab* was typed instead of *a* in the case of insertion

To get $P(Y|X)$, then, we find the probability of this kind of typo in this context. For insertion, for example (X_p is the p^{th} character of X):

(2) $P(Y|X) = \frac{ins[X_{p-1}, Y_p]}{count[X_{p-1}]}$

Some resources ...

Want to try these some of these things for yourself?

- *How to Write a Spelling Corrector* by Peter Norvig: http://norvig.com/spell-correct.html
  - 21 lines of Python code (other programming languages also available)
- Birkbeck spelling error corpus: http://www.ota.ox.ac.uk/headers/0643.xml

Spelling correction for web queries

A nice little side topic ...

Spelling correction for web queries is hard because it must handle:

- Proper names, new terms, etc. (*blog, shrek, nsync*)
- Frequent and severe spelling errors
- Very short contexts

Language and Computers
Writers' Aids
Introduction
Non-word error detection
Dictionaries
N-gram analysis
Isolated-word error correction
Types of errors
Rule-based methods
Similarity key techniques
Minimum edit distance
Probabilistic methods
Error correction for web queries
Grammar correction
Syntax and Computing
Grammar correction rules
Caveat emptor

Algorithm

Main Idea (Cucerzan and Brill (EMNLP-04))

- Iteratively transform the query into more likely queries
- Use query logs to determine likelihood
  - Despite the fact that many of these are misspelled!
  - Assumptions: the less wrong a misspelling is, the more frequent it is; and correct > incorrect

Example:

	anol scwartegger
→	*arnold schwartnegger*
→	*arnold schwarznegger*
→	*arnold schwarzenegger*

Language and Computers
Writers' Aids
Introduction
Non-word error detection
Dictionaries
N-gram analysis
Isolated-word error correction
Types of errors
Rule-based methods
Similarity key techniques
Minimum edit distance
Probabilistic methods
Error correction for web queries
Grammar correction
Syntax and Computing
Grammar correction rules
Caveat emptor

Algorithm (2)

- Compute the set of all *close* alternatives for each word in the query
  - Look at word unigrams and bigrams from the logs; this handles concatenation and splitting of words
  - Use weighted edit distance to determine closeness
- Search sequence of alternatives for best alternative string, using a noisy channel model

Constraint:

- No two adjacent in-vocabulary words can change simultaneously

Language and Computers
Writers' Aids
Introduction
Non-word error detection
Dictionaries
N-gram analysis
Isolated-word error correction
Types of errors
Rule-based methods
Similarity key techniques
Minimum edit distance
Probabilistic methods
Error correction for web queries
Grammar correction
Syntax and Computing
Grammar correction rules
Caveat emptor

Examples

Context Sensitivity

- *power crd* → *power cord*
- *video crd* → *video card*
- *platnuin rings* → *platinum rings*

Known Words

- *golf war* → *gulf war*
- *sap opera* → *soap opera*

Language and Computers
Writers' Aids
Introduction
Non-word error detection
Dictionaries
N-gram analysis
Isolated-word error correction
Types of errors
Rule-based methods
Similarity key techniques
Minimum edit distance
Probabilistic methods
Error correction for web queries
Grammar correction
Syntax and Computing
Grammar correction rules
Caveat emptor

Examples (2)

Tokenization

- *chat inspanich* → *chat in spanish*
- *ditroitigers* → *detroit tigers*
- *britenetspear inconcert* → *britney spears in concert*

Constraints

- *log wood* → *log wood* (not *dog food*)

Language and Computers
Writers' Aids
Introduction
Non-word error detection
Dictionaries
N-gram analysis
Isolated-word error correction
Types of errors
Rule-based methods
Similarity key techniques
Minimum edit distance
Probabilistic methods
Error correction for web queries
Grammar correction
Syntax and Computing
Grammar correction rules
Caveat emptor

Context-dependent word correction

Context-dependent word correction = correcting words based on the surrounding context.

- This will handle errors which are real words, just not the right one or not in the right form.
- This is very similar to a **grammar checker** = a mechanism which tells a user if their grammar is wrong.

Grammar correction—what does it correct?

- Syntactic errors = errors in how words are put together in a sentence: the order or form of words is incorrect, i.e., ungrammatical.
- **Local** syntactic errors: 1-2 words away
  - e.g., *The study was conducted mainly **be** John Black.*
  - A verb is where a preposition should be.
- **Long-distance** syntactic errors: (roughly) 3 or more words away
  - e.g., *The **kids** who are most upset by the little totem **is** going home early.*
  - Agreement error between subject *kids* and verb *is*

More on grammar correction

- Semantic errors = errors where the sentence structure sounds okay, but it doesn't really mean anything.
  - e.g., *They are leaving in about fifteen **minuets** to go to her house.*

  ⇒ *minuets* and *minutes* are both plural nouns, but only one makes sense here

There are many different ways in which grammar correctors work, two of which we'll focus on:

- *N*-gram model
- Rule-based model

N-gram grammar correctors

Remember that bigrams & trigrams model the probability of *sequences*

- Question *n*-grams address: Given the previous word (or two words), what is the probability of the current word?
- Use of *n*-grams: compare different candidates:
  - e.g., given *these*, we have a lower chance of seeing *report* than of seeing *reports*
  - Since a confusable word (*reports*) can be put in the same context, resulting in a higher probability, we flag *report* as a potential error

But there's a major problem: we may hardly ever see *these reports*, so we won't know its probability.

- **Some possible solutions:**
  - use bigrams/trigrams of **parts of speech**
  - use massive amounts of data and only flag errors when you have enough data to back it up

Rule-based grammar correctors

We can target specific error **patterns**. For example:

- *To a certain extend, we have achieved our goal.*
  1. Match the pattern *some* or *certain* followed by *extend*, which can be done using the **regular expression** `some|certain extend`
     - We'll discuss regular expressions with searching: for now, think of them as short ways to write patterns or templates
  2. Change the occurrence of *extend* in the pattern to *extent*.

See, e.g., http://www.languagetool.org/

Beyond regular expressions

- But what about correcting the following:
  - *A baseball teams were successful.*
- We should see that *A* is incorrect, but a simple pattern doesn't work because we don't know where the word *teams* might show up.
  - ***A** wildly overpaid, horrendous baseball **teams** were successful.* (Five words later; change needed.)
  - ***A** player on both my **teams** was successful.* (Five words later; no change needed.)
- We need to look at how the sentence is constructed in order to build a better rule.

Syntax

- **Syntax** = the study of the way that sentences are constructed from smaller units.
- There cannot be a “dictionary” for sentences since there is an infinite number of possible sentences:

(3) The house is large.

(4) John believes that the house is large.

(5) Mary says that John believes that the house is large.

There are two basic principles of sentence organization:

- Linear order
- Hierarchical structure (Constituency)

Linear order

- **Linear order** = the order of words in a sentence.
  - A sentence can have different meanings, based on its linear order:

(6) John loves Mary.

(7) Mary loves John.

  - Languages vary as to what extent this is true, but linear order in general is used as a guiding principle for organizing words into meaningful sentences.
- Simple linear order as such is not sufficient to determine sentence organization, though.
  - e.g., we can’t simply say “The verb is the second word in the sentence.”

(8) I **eat** at really fancy restaurants.

(9) Many executives **eat** at really fancy restaurants.

Language and Computers
Writers’ Aids
Introduction
Non-word error detection
Dictionaries
N-gram analysis
Isolated-word error correction
Types of errors
Rule-based methods
Similarity key techniques
Minimum edit distance
Probabilistic methods
Error correction for web queries
Grammar correction
Syntax and Computing
Grammar correction rules
Caveat emptor

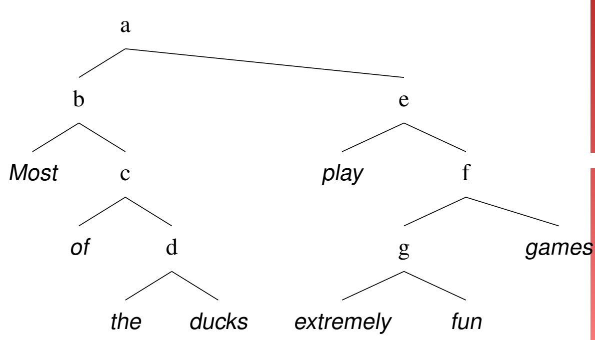
Constituency

- What are the “meaningful units” of a sentence like *Most of the ducks play extremely fun games*?
  - Most of the ducks
  - of the ducks
  - extremely fun
  - extremely fun games
  - play extremely fun games
- We refer to these meaningful groupings as **constituents** of a sentence.

Language and Computers
Writers’ Aids
Introduction
Non-word error detection
Dictionaries
N-gram analysis
Isolated-word error correction
Types of errors
Rule-based methods
Similarity key techniques
Minimum edit distance
Probabilistic methods
Error correction for web queries
Grammar correction
Syntax and Computing
Grammar correction rules
Caveat emptor

Hierarchical structure

- Constituents can appear within other constituents
- Constituents shown through brackets:
  [[Most [of [the ducks]]] [play [[extremely fun] games]]]
- Constituents displayed as a **syntactic tree**:



Language and Computers
Writers’ Aids
Introduction
Non-word error detection
Dictionaries
N-gram analysis
Isolated-word error correction
Types of errors
Rule-based methods
Similarity key techniques
Minimum edit distance
Probabilistic methods
Error correction for web queries
Grammar correction
Syntax and Computing
Grammar correction rules
Caveat emptor

Categories

- We would also like some way to say that
  - *the ducks*, and
  - *extremely fun games*

  are the same type of grouping, or constituent, whereas
  - *of the ducks*

  seems to be something else.
- For this, we will talk about different **categories**
  - Lexical
  - Phrasal

Language and Computers
Writers’ Aids
Introduction
Non-word error detection
Dictionaries
N-gram analysis
Isolated-word error correction
Types of errors
Rule-based methods
Similarity key techniques
Minimum edit distance
Probabilistic methods
Error correction for web queries
Grammar correction
Syntax and Computing
Grammar correction rules
Caveat emptor

Lexical categories

Lexical categories are simply word classes, or what you may have heard as **parts of speech**. The main ones are:

- verbs: *eat, drink, sleep, ...*
- nouns: *gas, food, lodging, ...*
- adjectives: *quick, happy, brown, ...*
- adverbs: *quickly, happily, well, westward*
- prepositions: *on, in, at, to, into, of, ...*
- determiners/articles: *a, an, the, this, these, some, much, ...*

Language and Computers
Writers’ Aids
Introduction
Non-word error detection
Dictionaries
N-gram analysis
Isolated-word error correction
Types of errors
Rule-based methods
Similarity key techniques
Minimum edit distance
Probabilistic methods
Error correction for web queries
Grammar correction
Syntax and Computing
Grammar correction rules
Caveat emptor

Determining lexical categories

How do we determine which category a word belongs to?

- **Distribution**: Where can these kinds of words appear in a sentence?
  - e.g., Nouns like *mouse* can appear after articles ("determiners") like *some*, while a verb like *eat* cannot.
- **Morphology**: What kinds of word prefixes/suffixes can a word take?
  - e.g., Verbs like *walk* can take a *ed* ending to mark them as past tense. A noun like *mouse* cannot.

(We'll discuss this more with Language Tutoring Systems)

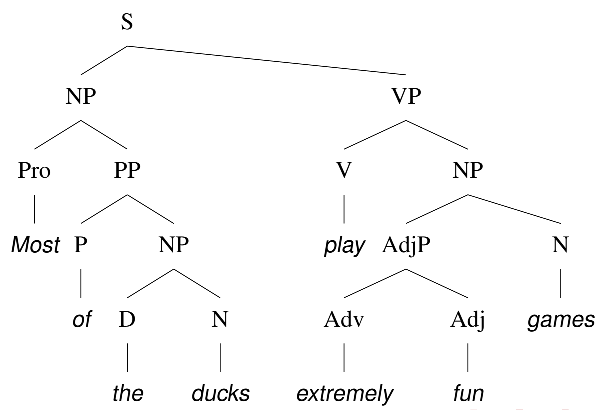
Phrasal categories

What about phrasal categories?

- What other phrases can we put in place of *The joggers* in a sentence such as the following?
  - The joggers ran through the park.
- Some options:
  - Susan
  - students
  - you
  - most dogs
  - some children
  - a huge, lovable bear
  - my friends from Brazil
  - the people that we interviewed
- Since all of these contain nouns, we consider these to be *noun phrases*, abbreviated with NP.

Building a tree

Other phrases work similarly (S = sentence, VP = verb phrase, PP = prepositional phrase, AdjP = adjective phrase):



Phrase Structure Rules

- We can give rules for building these phrases.
  - We want a way to say that a determiner and a noun make up a noun phrase, but a verb and an adverb do not.
- **Phrase structure rules** are a way to build larger constituents from smaller ones.
  - e.g., S → NP VP
    This says:
    - A sentence (S) constituent is composed of a noun phrase (NP) constituent and a verb phrase (VP) constituent. [hierarchy]
    - The NP must precede the VP. [linear order]

Some other possible English rules

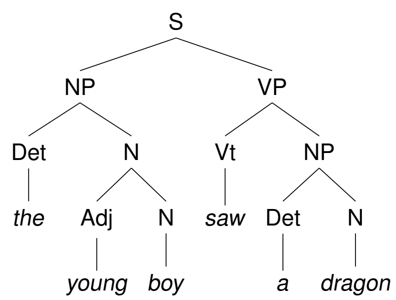
- NP → Det N (*the cat, a house, this computer*)
- NP → Det AdjP N (*the happy cat, a really happy house*)
  - For phrase structure rules, as shorthand parentheses are used to express that a category is optional.
  - We thus can compactly express the two rules above as one rule: NP → Det (AdjP) N
- AdjP → (Adv) Adj (*really happy*)
- VP → V (*laugh, run, eat*)
- VP → V NP (*love John, hit the wall, eat cake*)
- VP → V NP NP (*give John the ball*)
- PP → P NP (*to the store, at John, in a New York minute*)
- NP → NP PP (*the cat on the stairs*)

Phrase Structure Rules and Trees

With every phrase structure rule, you can draw a tree for it.

Lexicon:
Vt → *saw*
Det → *the*
Det → *a*
N → *dragon*
N → *boy*
Adj → *young*

Syntactic rules:
S → NP VP
VP → Vt NP
NP → Det N
N → Adj N



Some Properties of Phrase Structure Rules

- Potentially **(structurally) ambiguous** = have more than one analysis

(10) We need more intelligent leaders.

(11) Paraphrases:

a. We need leaders who are more intelligent.

b. Intelligent leaders? We need more of them!

- **Recursive** = property allowing for a rule to be reapplied (within its hierarchical structure).
  - e.g., **NP** → **NP** PP
  - The property of recursion means that the set of potential sentences in a language is **infinite**.

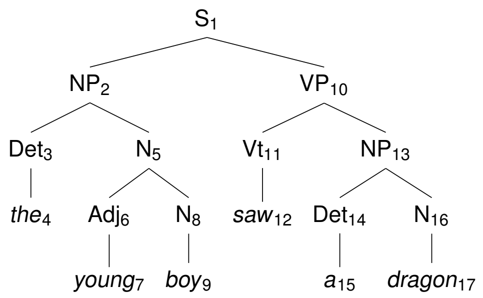
Parsing

Using these phrase structure rules, we can get a computer to **parse** a sentence = assign a structure to a sentence.

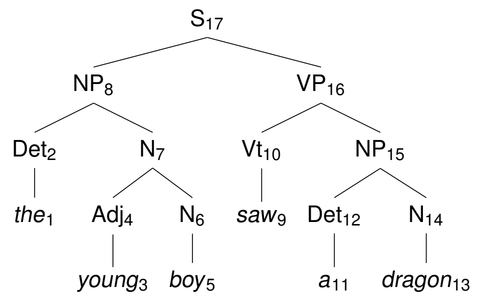
There are many, many parsing techniques out there.

- **Top-down**: build a tree by starting at the top (i.e. S → NP VP) and working down the tree.
- **Bottom-up**: build a tree by starting with the words at the bottom and working up to the top.

Trace of a top-down parse



Trace of a bottom-up parse



More finely articulated rules

In practice, one actually works with rules like:

- $S \rightarrow NP_{pl}\ VP_{pl}$

Or uses features & variables like:

- $S \rightarrow NP_{NUM=X}\ VP_{NUM=X}$

It can get very complicated (& fun) very quickly:

- $S_{TENSE=Z} \rightarrow NP_{NUM=X,PER=Y}\ VP_{NUM=X,PER=Y,TENSE=Z}$

Writing grammar correction rules

So, with our rules, we can now write some correction rules, which we will just sketch here.

- *A baseball teams were successful.*
  - *A* followed by PLURAL NP: change *A* → *The*
  - i.e., one looks for a tree like: NP → Det_{sg} NP_{pl}
    - We'll talk about this more with mal-rules in Language Tutoring Systems
- *John at the pizza.*
  - The structure of this sentence is NP PP, but that doesn't make up a whole sentence.
  - We need a verb somewhere.

Dangers of spelling and grammar correction

- The more we depend on spelling correctors, do we try less to correct things on our own?
  - But spell checkers are not 100%
- One (older) study found that students made **more** errors (in proofreading) when using a spell checker!

	high SAT scores	low SAT scores
use checker	16 errors	17 errors
no checker	5 errors	12.3 errors

(cf., http://www.wired.com/news/business/0,1367,58058,00.html)

Candidate for a Pullet Surprise

(“The Spell-Checker Poem”)

by Mark Eckman and Jerrold H. Zar
http://grammar.about.com/od/spelling/a/spellcheck.htm

I have a spelling checker,
It came with my PC.
It plane lee marks four my revue
Miss steaks aye can knot sea.

Eye ran this poem threw it,
Your sure reel glad two no.
Its vary polished in it's weigh.
My checker tolled me sew.

...

Language and Computers
Writers' Aids
Introduction
Non-word error detection
Dictionaries
N-gram analysis
Isolated-word error correction
Types of errors
Rule-based methods
Similarity key techniques
Minimum edit distance
Probabilistic methods
Error correction for web queries
Grammar correction
Syntax and Computing
Grammar correction rules
Caveat emptor

References

- The discussion is based on Markus Dickinson (2006). Writer's Aids. In Keith Brown (ed.): *Encyclopedia of Language and Linguistics. Second Edition.*. Elsevier.
- A major inspiration for that article and our discussion is Karen Kukich (1992): *Techniques for Automatically Correcting Words in Text.* ACM Computing Surveys, pages 377–439; as well as Roger Mitton (1996), *English Spelling and the Computer.*
- For a discussion of the confusion matrix, cf. Mark D. Kernighan, Kenneth W. Church and William A. Gale (1990). A spelling Correction Program Based on a Noisy Channel Model. In *Proceedings of COLING-90.* pp. 205–210.
- An open-source style/grammar checker is described in Daniel Naber (2003). *A Rule-Based Style and Grammar Checker.* Diploma Thesis, Universität Bielefeld. https://www.languagetool.org

Language and Computers
Writers' Aids
Introduction
Non-word error detection
Dictionaries
N-gram analysis
Isolated-word error correction
Types of errors
Rule-based methods
Similarity key techniques
Minimum edit distance
Probabilistic methods
Error correction for web queries
Grammar correction
Syntax and Computing
Grammar correction rules
Caveat emptor