Statistical Machine Translation

Bonnie Dorr   Christof Monz

CMSC 723: Introduction to Computational Linguistics

Lecture 8

October 27, 2004

Overview

- Why MT
- Statistical vs. rule-based MT
- Computing translation probabilities from a parallel corpus
- IBM Models 1-3

A Brief History

- Machine translation was one of the first applications envisioned for computers
- Warren Weaver (1949): "I have a text in front of me which is written in Russian but I am going to pretend that it is really written in English and that it has been coded in some strange symbols. All I need to do is strip off the code in order to retrieve the information contained in the text."
- First demonstrated by IBM in 1954 with a basic word-for-word translation system

Interest in MT

- Commercial interest:
  - U.S. has invested in MT for intelligence purposes
  - MT is popular on the web—it is the most used of Google's special features
  - EU spends more than $1 billion on translation costs each year.
  - (Semi-)automated translation could lead to huge savings

Interest in MT

- Academic interest:
  - One of the most challenging problems in NLP research
  - Requires knowledge from many NLP sub-areas, e.g., lexical semantics, parsing, morphological analysis, statistical modeling,…
  - Being able to establish links between two languages allows for transferring resources from one language to another

Rule-Based vs. Statistical MT

- Rule-based MT:
  - Hand-written transfer rules
  - Rules can be based on lexical or structural transfer
  - Pro: firm grip on complex translation phenomena
  - Con: Often very labor-intensive -> lack of robustness
- Statistical MT
  - Mainly word or phrase-based translations
  - Translation are learned from actual data
  - Pro: Translations are learned automatically
  - Con: Difficult to model complex translation phenomena

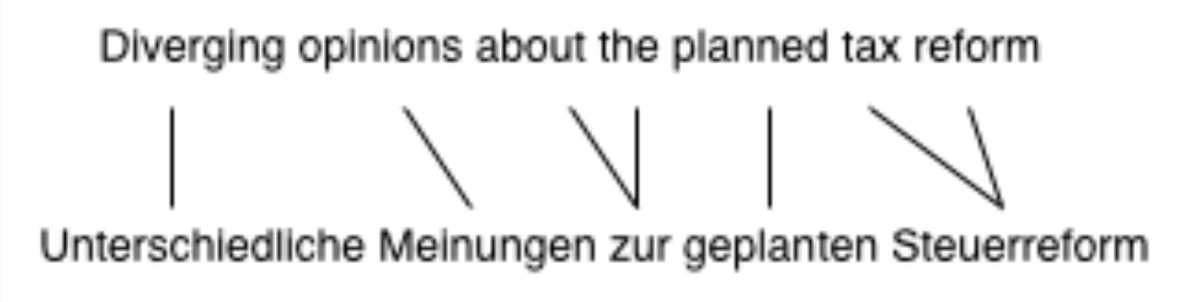
Parallel Corpus

- Example from DE-News (8/1/1996)

English	German
Diverging opinions about planned tax reform	Unterschiedliche Meinungen zur geplanten Steuerreform
The discussion around the envisaged major tax reform continues .	Die Diskussion um die vorgesehene grosse Steuerreform dauert an .
The FDP economics expert , Graf Lambsdorff , today came out in favor of advancing the enactment of significant parts of the overhaul , currently planned for 1999 .	Der FDP - Wirtschaftsexperte Graf Lambsdorff sprach sich heute dafuer aus , wesentliche Teile der fuer 1999 geplanten Reform vorzuziehen .

Word-Level Alignments

- Given a parallel sentence pair we can link (align) words or phrases that are translations of each other:



Parallel Resources

- Newswire: DE-News (German-English), Hong-Kong News, Xinhua News (Chinese-English),
- Government: Canadian-Hansards (French-English), Europarl (Danish, Dutch, English, Finnish, French, German, Greek, Italian, Portugese, Spanish, Swedish), UN Treaties (Russian, English, Arabic, . . .)
- Manuals: PHP, KDE, OpenOffice (all from OPUS, many languages)
- Web pages: STRAND project (Philip Resnik)

Sentence Alignment

- If document D_e is translation of document D_f how do we find the translation for each sentence?
- The *n*-th sentence in D_e is not necessarily the translation of the *n*-th sentence in document D_f
- In addition to 1:1 alignments, there are also 1:0, 0:1, 1:n, and n:1 alignments
- Approximately 90% of the sentence alignments are 1:1

Sentence Alignment (c’ntd)

- There are several sentence alignment algorithms:
  - Align (Gale & Church): Aligns sentences based on their character length (shorter sentences tend to have shorter translations then longer sentences). Works astonishingly well
  - Char-align: (Church): Aligns based on shared character sequences. Works fine for similar languages or technical domains
  - K-Vec (Fung & Church): Induces a translation lexicon from the parallel texts based on the distribution of foreign-English word pairs.

Computing Translation Probabilities

- Given a parallel corpus we can estimate P(e | f) The maximum likelihood estimation of P(e | f) is: freq(e,f)/freq(f)
- Way too specific to get any reasonable frequencies! Vast majority of unseen data will have zero counts!
- P(e | f) could be re-defined as:

$$P(e \mid f) = \prod_{f_j} \max_{e_i} P(e_i \mid f_j)$$

- Problem: The English words maximizing P(e | f) might not result in a readable sentence

Computing Translation Probabilities (c’ tnd)

- We can account for adequacy: each foreign word translates into its most likely English word
- We cannot guarantee that this will result in a fluent English sentence
- Solution: transform P(e | f) with Bayes’ rule: $P(e \mid f) = P(e)\, P(f \mid e) / P(f)$
- $P(f \mid e)$ accounts for adequacy
- $P(e)$ accounts for fluency

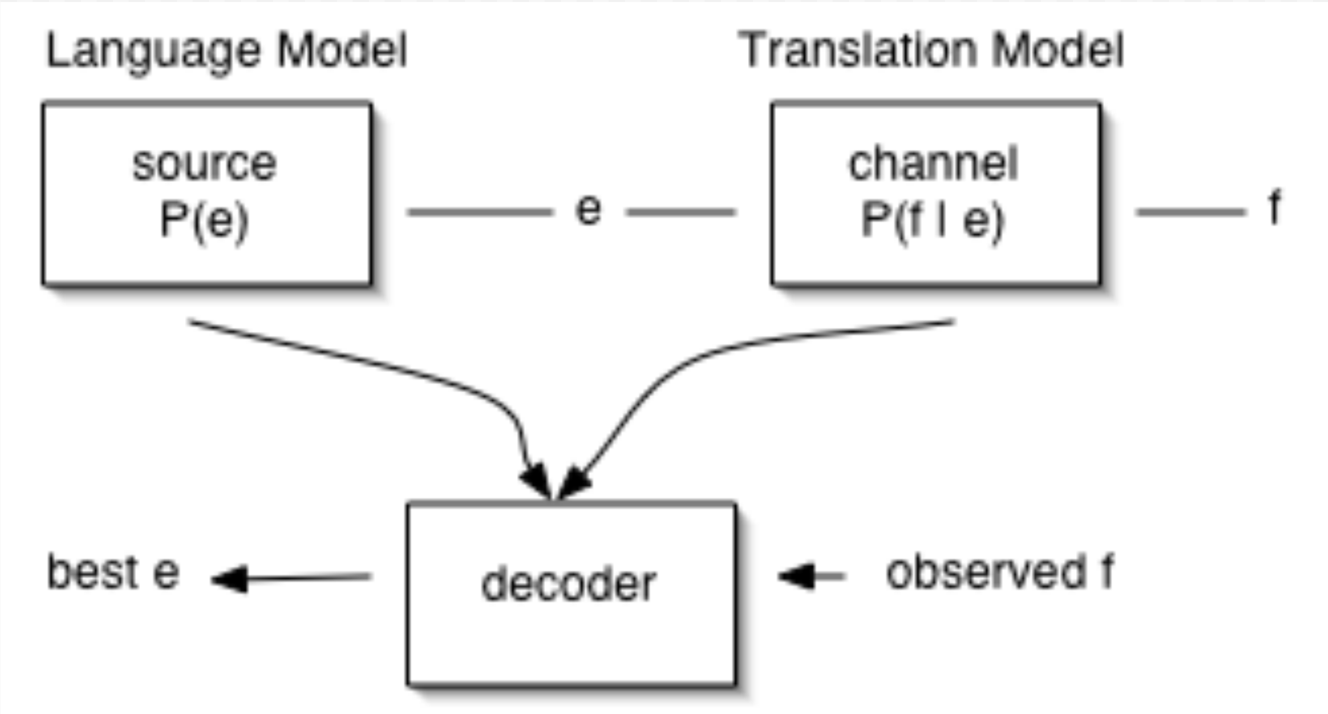
Decoding

- The decoder combines the evidence from P(e) and P(f | e) to find the sequence e that is the best translation:

$$\arg\max_{e} P(e \mid f) = \arg\max_{e} P(f \mid e)P(e)$$

- The choice of word e' as translation of f' depends on the translation probability P(f' | e') and on the context, i.e. other English words preceding e'

Noisy Channel Model for Translation



Language Modeling

- Determines the probability of some English sequence e_1^l of length l
- P(e) is hard to estimate directly, unless l is very small

$$P(e_1^l) = P(e_1)\prod_{i=2}^{l} P(e_i \mid e_1^{i-1})$$

- P(e) is normally approximated as:

$$P(e_1^l) = P(e_1)P(e_2 \mid e_1)\prod_{i=3}^{l} P(e_i \mid e_{i-m}^{i-1})$$

where m is size of the context, i.e. number of previous words that are considered, normally m=2 (tri-gram language model

Translation Modeling

- Determines the probability that the foreign word f is a translation of the English word e
- How to compute P(f | e) from a parallel corpus?
- Statistical approaches rely on the co-occurrence of e and f in the parallel data: If e and f tend to co-occur in parallel sentence pairs, they are likely to be translations of one another

Finding Translations in a Parallel Corpus

- Into which foreign words f, . . . , f' does e translate?
- Commonly, four factors are used:
  - How often do e and f co-occur? (translation)
  - How likely is a word occurring at position i to translate into a word occurring at position j? (distortion) For example: English is a verb-second language, whereas German is a verb-final language
  - How likely is e to translate into more than one word? (fertility) For example: *defeated* can translate into *eine Niederlage erleiden*
  - How likely is a foreign word to be spuriously generated? (null translation)

Translation Steps

IBM Models 1–5

- Model 1: Bag of words
  - Unique local maxima
  - Efficient EM algorithm (Model 1–2)
- Model 2: General alignment: $a(e_{pos} \mid f_{pos}, e_{length}, f_{length})$
- Model 3: fertility: n(k l e)
  - No full EM, count only neighbors (Model 3–5)
  - Deficient (Model 3–4)
- Model 4: Relative distortion, word classes
- Model 5: Extra variables to avoid deficiency

IBM Model 1

- Given an English sentence $e_1 \ldots e_l$ and a foreign sentence $f_1 \ldots f_m$
- We want to find the 'best' alignment a, where a is a set pairs of the form {(i , j), . . . , (i', j')},

  $0 <= i , i' <= l$ and $1 <= j , j' <= m$
- Note that if (i , j), (i', j) are in a, then i equals i', i.e. no many-to-one alignments are allowed
- Note we add a spurious NULL word to the English sentence at position 0
- In total there are $(l + 1)^m$ different alignments A
- Allowing for many-to-many alignments results in $(2^l)^m$ possible alignments A

IBM Model 1

- Simplest of the IBM models
- Does not consider word order (bag-of-words approach)
- Does not model one-to-many alignments
- Computationally inexpensive
- Useful for parameter estimations that are passed on to more elaborate models

IBM Model 1

- Translation probability in terms of alignments:

$$P(f \mid e) = \sum_{a \in A} P(f, a \mid e)$$

where:

$$P(f, a \mid e) = P(a \mid e) \cdot P(f \mid a, e)$$

$$= \frac{1}{(l+1)^m} \prod_{j=1}^{m} P(f_j \mid e_{a_j})$$

and:

$$P(f \mid e) = \sum_{a \in A} \frac{1}{(l+1)^m} \prod_{j=1}^{m} P(f_j \mid e_{a_j})$$

IBM Model 1

- We want to find the most likely alignment:

$$\arg\max_{a \in A} \frac{1}{(l+1)^m} \prod_{j=1}^{m} P(f_j \mid e_{a_j})$$

- Since P(a | e) is the same for all a:

$$\arg\max_{a \in A} \prod_{j=1}^{m} P(f_j \mid e_{a_j})$$

- Problem: We still have to enumerate all alignments

IBM Model 1

- Since $P(f_j \mid e_i)$ is independent from $P(f_{j'} \mid e_{i'})$ we can find the maximum alignment by looking at the individual translation probabilities only
- Let $\arg\max_{a \in A} = (a_1, \ldots, a_m)$, then for each a_j:

$$a_j = \arg\max_{0 \le i \le l} P(f_j \mid e_i)$$

- The best alignment can computed in a quadratic number of steps: (l+1 x m)

Computing Model 1 Parameters

- How to compute translation probabilities for model 1 from a parallel corpus?
- Step 1: Determine candidates. For each English word e collect all foreign words f that co-occur at least once with e
- Step 2: Initialize P(f | e) uniformly, i.e.
  - P(f | e) = 1/(no of co-occurring foreign words)

Computing Model 1 Parameters

- Step 3: Iteratively refine translation probablities:

```
for n iterations
    set tc to zero
    for each sentence pair (e,f) of lengths (l,m)
        for j=1 to m
            total=0;
            for i=1 to l
                total += P(f_j | e_i);
            for i=1 to l
                tc(f_j | e_i) += P(f_j | e_i)/total;
    for each word e
        total=0;
        for each word f s.t. tc(f | e) is defined
            total += tc(f | e);
        for each word f s.t. tc(f | e) is defined
            P(f | e) = tc(f | e)/total;
```

IBM Model 1 Example

- Parallel 'corpus':

  the dog :: le chien

  the cat :: le chat

- Step 1+2 (collect candidates and initialize uniformly):

  P(le | the) = P(chien | the) = P(chat | the) = 1/3

  P(le | dog) = P(chien | dog) = P(chat | dog) = 1/3

  P(le | cat) = P(chien | cat) = P(chat | cat) = 1/3

  P(le | NULL) = P(chien | NULL) = P(chat | NULL) = 1/3

IBM Model 1 Example

- Step 3: Iterate
- NULL the dog :: le chien
  - j=1

    total = P(le | NULL)+P(le | the)+P(le | dog)= 1

    tc(le | NULL) += P(le | NULL)/1 = 0 += .333/1 = 0.333

    tc(le | the) += P(le | the)/1 = 0 += .333/1 = 0.333

    tc(le | dog) += P(le | dog)/1 = 0 += .333/1 = 0.333
  - j=2

    total = P(chien | NULL)+P(chien | the)+P(chien | dog)=1

    tc(chien | NULL) += P(chien | NULL)/1 = 0 += .333/1 = 0.333

    tc(chien | the) += P(chien | the)/1 = 0 += .333/1 = 0.333

    tc(chien | dog) += P(chien | dog)/1 = 0 += .333/1 = 0.333

IBM Model 1 Example

- NULL the cat :: le chat
  - j=1
    total = P(le | NULL)+P(le | the)+P(le | cat)=1
    tc(le | NULL) += P(le | NULL)/1 = 0.333 += .333/1 = 0.666
    tc(le | the) += P(le | the)/1 = 0.333 += .333/1 = 0.666
    tc(le | cat) += P(le | cat)/1 = 0 +=.333/1 = 0.333
  - j=2
    total = P(chien | NULL)+P(chien | the)+P(chien | dog)=1
    tc(chat | NULL) += P(chat | NULL)/1 = 0 += .333/1 = 0.333
    tc(chat | the) += P(chat | the)/1 = 0 += .333/1 = 0.333
    tc(chat | cat) += P(chat | dog)/1 = 0 += .333/1 = 0.333

IBM Model 1 Example

- Re-compute translation probabilities
  - total(the) = tc(le I the) + tc(chien I the) + tc(chat I the)
    = 0.666 + 0.333 + 0.333 = 1.333
    P(le I the) = tc(le I the)/total(the)
    = 0.666 / 1.333 = 0.5
    P(chien I the) = tc(chien I the)/total(the)
    = 0.333/1.333 0.25
    P(chat I the) = tc(chat I the)/total(the)
    = 0.333/1.333 0.25
  - total(dog) = tc(le I dog) + tc(chien I dog) = 0.666
    P(le I dog) = tc(le I dog)/total(dog)
    = 0.333 / 0.666 = 0.5
    P(chien I dog) = tc(chien I dog)/total(dog)
    = 0.333 / 0.666 = 0.5

IBM Model 1 Example

- Iteration 2:
- NULL the dog :: le chien
  - j=1

    total = P(le | NULL)+P(le | the)+P(le | dog)= 1.5

    = 0.5 + 0.5 + 0.5 = 1.5

    tc(le | NULL) += P(le | NULL)/1 = 0 += .5/1.5 = 0.333

    tc(le | the) += P(le | the)/1 = 0 += .5/1.5 = 0.333

    tc(le | dog) += P(le | dog)/1 = 0 += .5/1.5 = 0.333
  - j=2

    total = P(chien | NULL)+P(chien | the)+P(chien | dog)=1

    = 0.25 + 0.25 + 0.5 = 1

    tc(chien | NULL) += P(chien | NULL)/1 = 0 += .25/1 = 0.25

    tc(chien | the) += P(chien | the)/1 = 0 += .25/1 = 0.25

    tc(chien | dog) += P(chien | dog)/1 = 0 += .5/1 = 0.5

IBM Model 1 Example

- NULL the cat :: le chat
  - j=1

    total = P(le | NULL)+P(le | the)+P(le | cat)= 1.5

    = 0.5 + 0.5 + 0.5 = 1.5

    tc(le | NULL) += P(le | NULL)/1 = 0.333 += .5/1 = 0.833

    tc(le | the) += P(le | the)/1 = 0.333 += .5/1 = 0.833

    tc(le | cat) += P(le | cat)/1 = 0 += .5/1 = 0.5
  - j=2

    total = P(chat | NULL)+P(chat | the)+P(chat | cat)=1

    = 0.25 + 0.25 + 0.5 = 1

    tc(chat | NULL) += P(chat | NULL)/1 = 0 += .25/1 = 0.25

    tc(chat | the) += P(chat | the)/1 = 0 += .25/1 = 0.25

    tc(chat | cat) += P(chat | cat)/1 = 0 += .5/1 = 0.5

IBM Model 1 Example

- Re-compute translations (iteration 2):
  - total(the) = tc(le | the) + tc(chien | the) + tc(chat | the)

    = .833 + 0.25 + 0.25 = 1.333

    P(le | the) = tc(le | the)/total(the)

    = .833 / 1.333 = 0.625

    P(chien | the) = tc(chien | the)/total(the)

    = 0.25/1.333 = 0.188

    P(chat | the) = tc(chat | the)/total(the)

    = 0.25/1.333 = 0.188
  - total(dog) = tc(le | dog) + tc(chien | dog)

    = 0.333 + 0.5 = 0.833

    P(le | dog) = tc(le | dog)/total(dog)

    = 0.333 / 0.833 = 0.4

    P(chien | dog) = tc(chien | dog)/total(dog)

    = 0.5 / 0.833 = 0.6

IBM Model 1Example

- After 5 iterations:

P(le I NULL) = 0.755608028335301

P(chien I NULL) = 0.122195985832349

P(chat I NULL) = 0.122195985832349

P(le I the) = 0.755608028335301

P(chien I the) = 0.122195985832349

P(chat I the) = 0.122195985832349

P(le I dog) = 0.161943319838057

P(chien I dog) = 0.838056680161943

P(le I cat) = 0.161943319838057

P(chat I cat) = 0.838056680161943

IBM Model 1 Recap

- IBM Model 1 allows for an efficient computation of translation probabilities
- No notion of fertility, i.e., it's possible that the same English word is the best translation for all foreign words
- No positional information, i.e., depending on the language pair, there might be a tendency that words occurring at the beginning of the English sentence are more likely to align to words at the beginning of the foreign sentence

IBM Model 3

- IBM Model 3 offers two additional features compared to IBM Model 1:
  - How likely is an English word e to align to k foreign words (fertility)?
  - Positional information (distortion), how likely is a word in position i to align to a word in position j?

IBM Model 3: Fertility

- The best Model 1 alignment could be that a single English word aligns to all foreign words
- This is clearly not desirable and we want to constrain the number of words an English word can align to
- Fertility models a probability distribution that word e aligns to k words: n(k,e)
- Consequence: translation probabilities cannot be computed independently of each other anymore
- IBM Model 3 has to work with full alignments, note there are up to $(l+1)^m$ different alignments

IBM Model 1 + Model 3

- Iterating over all possible alignments is computationally infeasible
- Solution: Compute the best alignment with Model 1 and change some of the alignments to generate a set of likely alignments (pegging)
- Model 3 takes this restricted set of alignments as input

Pegging

- Given an alignment a we can derive additional alignments from it by making small changes:
  - Changing a link (j,i) to (j,i')
  - Swapping a pair of links (j,i) and (j',i') to (j,i') and (j',i)
- The resulting set of alignments is called the neighborhood of a

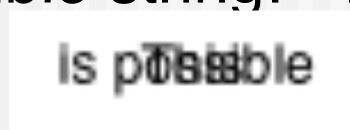
IBM Model 3: Distortion

- The distortion factor determines how likely it is that an English word in position i aligns to a foreign word in position j, given the lengths of both sentences:

$$d(j \mid i, l, m)$$

- Note, positions are absolute positions

Deficiency

- Problem with IBM Model 3: It assigns probability mass to impossible strings
  - Well formed string: “This is possible”
  - Ill-formed but possible string: “This possible is”
  - Impossible string: 
- Impossible strings are due to distortion values that generate different words at the same position
- Impossible strings can still be filtered out in later stages of the translation process

Limitations of IBM Models

- Only 1-to-N word mapping
- Handling fertility-zero words (difficult for decoding)
- Almost no syntactic information
  - Word classes
  - Relative distortion
- Long-distance word movement
- Fluency of the output depends entirely on the English language model

Decoding

- How to translate new sentences?
- A decoder uses the parameters learned on a parallel corpus
  - Translation probabilities
  - Fertilities
  - Distortions
- In combination with a language model the decoder generates the most likely translation
- Standard algorithms can be used to explore the search space (A*, greedy searching, …)
- Similar to the traveling salesman problem