APPLICATIONS 1:

MACHINE TRANSLATION II

Theme

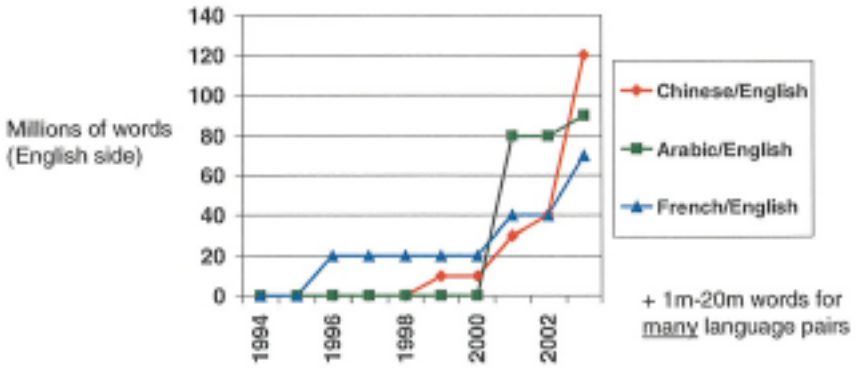
The second of two lectures on Machine Translation. Developments in MT since 1990: statistical training of MT systems to overcome the human effort bottleneck (CANDIDE and EGYPT). Systematically up the Vauqois triangle.

Summary of Contents

1. Background

History: MT as an instance of decoding/cryptanalysis: Warren Weaver (1949) to Claude Shannon. Tried in 1950s. Now: faster computers (100,000×), larger computer memory, more text, online dictionaries, work in speech recognition and lexicography and NLP...

Growth of online text:



(Data stripped of formatting, in sentence-pair format, available from the Linguistic Data Consortium at UPenn).

2. Statistical MT: CANDIDE

Approach: In the late 1980's, IBM's CANDIDE group took Canadian Parliamentary Hansard (3 mill sentences, French and English). They say French is just garbled English, so ungarble it!

General approach is to build two models and then, given a sentence, 'decode' it by finding the most probable words and rules in two decoding steps:

1. Translation model: Find best target-language word or phrase for each input unit. Result: a bag of words/phases
2. Language model: Pick one word/phrase at a time from the bag of words/phrases and build up a sentence

Language model: bigrams (and later, trigrams) of English, from Wall Street Journal, with frequency counts normalized to probabilities

Translation model: try series of increasingly complex translation models:

- Model 1: word/phrase alignment: models at first assume simple word-word and phrase-phrase correspondence
- Model 2: fertility: how many word(s) a single word translates into (for French and English, it's mostly one-to-one)
- Model 3: distortion: changes in absolute word position (French and English have more or less same word order)
- Model 4: relative word group distortion

Theory: The Noisy Channel Model, using Bayes' Theorem

$$P(y|x).P(x) = P(x|y).P(y)$$

rephrased, for French and English, as

$$P(E|F).P(F) = P(F|E).P(E)$$

where $P(F|E)$ is the translation model and $P(E)$ is the language model. Since $P(F)$ is constant for a given sentence, we can ignore it. Then $P(E|F)$ (probability that the input French gives this particular sentence of English) = $P(F|E)$ (probability that the speaker wanted to say it that way, given what I know of the way English sentences are and how they go to French) × $P(E)$ (probability governing how the English words are properly composed, for which there is plenty of training material). Result: what IBM in their humility called the 'fundamental equation of MT':

$$P(E|F) = argmax\ P(F|E).P(E)$$

That is, for a given French sentence, maximize the arguments (the words and ordering, etc.) of this equation, to get the best English equivalent.

Why 'turn it around' this way? Because then you can use $P(E)$, instead of $P(F)$, and there is a lot more data available online for English than for French (or some other language you want to translate from).

These two models in practice consist of sets of tables, listing the probability values of certain parameter (combinations).

Constructing the Translation Model:

The problem is to find all French word(s) that translate each English word, together with their (normalized relative to one another) statistics. For this, they used the Expectation Maximization (EM) algorithm, something developed and used a long time ago by statisticians (Dempster et al. 1977).

Example of EM algorithm (from Knight 1999 SMT Tutorial booklet)

Imagine you have two translated sentence pairs:

S1 English: a b French: x y
S2 English: a French: y

Not knowing anything more, you say

t(x|a) = ½ t(y|a) = ½ t(x|b) = ½ t(y|b) = ½

That is, word x can come from a or from b, and y too, equally probably. So you have three models a1 (a↔x and b↔y) and a2 (a↔y and b↔x) from sentence S1 and a3 (a↔y) from sentence S2.

Step 1: Compute the overall likelihood of each translation model:
P(a1, f|e) = ½ * ½ = ¼ from the a↔x and b↔y respectively for S1
P(a2, f|e) = ½ * ½ = ¼ from the a↔y and b↔x respectively for S1
P(a3, f|e) = ½ from the a↔y for S2

Step 2: Normalize these models out:
P(a1 | e,f) = ¼ / (¼ + ¼) = ½ from a1 compared to a1 and a2 together
P(a2 | e,f) = ¼ / (¼ + ¼) = ½ from a2 compared to a1 and a2 together
P(a3 | e,f) = ½ / ½ = 1 from a3 compared to a3 all by itself (there's only one possibility)

Step 3: Now consider the individual terms:
t(x|a) = ½ out of model a1
t(y|a) = ½ + 1 = 3/2 out of models a1 + a3 (that's all the evidence we have for y and a, from all models combined)
t(x|b) = ½ out of model a2
t(y|b) = ½ also out of a2

Step 4: Again, normalize (but now looking at the terms, not the models):
t(x|a) = ½ / (½ + 3/2) = ¼ that's x|a's proportion of the two possible productions out of a
t(y|a) = 3/2 / (½ + 3/2) = ¾ and y|a's proportion of out a
t(x|b) = ½ / (½ + ½) = ½ and the same for x|b out of b
t(y|b) = ½ / (½ + ½) = ½ and for y|b from b

Step 5: Now we go back to the models, like in step 1, with this new information (before step 1 we started with everything equal; now the probabilities have shifted a little because of S2):
P(a1, f|e) = ¼ * ½ = 1/8 from the a↔x and b↔y respectively
P(a2, f|e) = 3/4 * ½ = 3/8 from the a↔y and b↔x respectively
P(a3, f|e) = 3/4 from the a↔y

Step 6: This is not normalized, so we normalize again, like step 2:
P(a1 | e,f) = 1/8 / (1/8 + 3/8) = ¼ from a1 compared to a1 and a2
P(a2 | e,f) = 3/8 / (1/8 + 3/8) = 3/4 from a2 compared to a1 and a2
P(a3 | e,f) = ¾ / ¾ = 1 from a3 compared to a3 all by itself (there's only one possibility)

Step 7: Repeat step 3 to compute the terms' scores:
t(x|a) = ¼ from a1 (that's all we know about x coming from a)
t(y|a) = ¼ + 1 = 7/4 from a1 and a3 (that's the info we have about y coming from a!)
t(x|b) = ¾ from a2
t(y|b) = ¼ ditto

Step 8: Now again normalize the terms' scores, as in step 4:
t(x|a) = ¼ / (¼ + 7/4) = 1/8

t(y|a) = 7/4 / (¼ + 7/4) = 7/8
t(x|b) = ¾ / (¾ + ¼) = ¾
t(y|b) = ¼ / (¾ + ¼) = ¼

Looks what's happened. The 'probability mass' is gradually being pulled toward models a2, thanks to the initial asymmetry introduced by Sentence S2. If you repreat the above process a long time, you get

t(x|a) = ¼ / (¼ + 7/4) = 0.0000001
t(y|a) = 7/4 / (¼ + 7/4) = 0.9999999
t(x|b) = ¾ / (¾ + ¼) = 0.9999999
t(y|b) = ¼ / (¾ + ¼) = 0.0000001

EM allows you to make maximum likelihood estimates over a large collection of observations, proportionally to the sample you have seen.

Alignments

To now use this, you start with a large collection of parallel sentences in English and French. You draw a line from each French word to all possible English words it could translate to, in all English sentences whose French translation contains your French word. And then you set EM loose!

I. For each French word f,

1. find all French sentences F containing f,
2. find all the corresponding English sentences E,
3. list all possible links from each word in each F_i to the word(s) in E_i (choosing increasing fertilities: 0, 1, 2 ... 25)
4. record the word–word(s) pairs
5. repeat this, counting the number of times you obtain each pair

II. For each sentence pair, select a complete set of links for all French words so that no English words are left unlinked, and compute a goodness score for this based on the frequency of occurrence of the units. Do so for all possible complete sets of links. Normalize these scores to obtain true probabilities.

III Result: A translation table that lists the probability of each word on one side being translated to its equivalent(s) on the other.

Since this didn't quite work — IBM noticed that some words in French correspond to zero, or to two , words in English, they also computer a 'fertility table' that listed how likely each English word was to have zero, one, or two French words in translation.

And later, they built a table to indicate how likely a French word was to move from the position its English equivalent was in in the sentence (the 'distortion table').

So they built several tables:

1. Translation word table: *P(f|e)*: English-French word pairs
2. Fertility table: *P(n|e),* 0 <= n <= 25: # English words each French word corresponds to
3. Distortion table: *P(i|j,l)* for positions *i* and *j* and sent length *l* <= 25: word movement

English: the				English: not				English: hear			
French	Prob	Fertility	Prob	French	Prob	Fertility	Prob	French	Prob	Fertility	Prob
le	0.610	1	0.871	pas	0.469	2	0.758	bravo	0.992	0	0.584
la	0.178	0	0.124	ne	0.460	0	0.133	entendre	0.005	1	0.416
l'	0.083	2	0.004	non	0.024	1	0.106	entendu	0.002		
les	0.023			Faux	0.003			entends	0.001		
il	0.012			jamais	0.002						
de	0.009										

From this table, you can see that "the" corresponds to "le" with probability 0.610, with "la" with probability 0.124, and so on. Also, "not" corresponds to two words (fertility 2 with highest probability), namely "pas" and "ne", both about equally. Can you see what happens in the last example, for "hear"?

NOTE: IBM did not develop all this in one shot. Initially they simple used (in what they called 'Model 1') word–word correspondences. Then they noticed that some words have multi-word translation, and so added (in Model 2) the fertility tables that specified how likely a word was to translate into zero, one, two, or more words. In Model 3 they tried to constrain translation words to stay more or less in the same sequence as the input words, using the distortion table. (This would not work for some other language pairs, of course). They added further refinements to two later models.

Constructing the Language Model: Collect all trigrams from English corpus. Used lots of Wall Street Journal to obtain a (massive) table of all the two-word (and later, three-word) sequences seen (e.g., you never see "the the a" but you might once or twice have seen "the big man").

Test language model: scrambled 38 English sentences (max length 11 words), ran through language model, computing highest-probability word sequence. Got exact match 63%, close match 21%, junk 16%.

Translating: Now comes the actual translation. This is cast as a search problem to optimize the best combination of translation words, in sequence.

Decoding: given input sentence f, find best English equivalent e. Used stack search (Bahl et al.):

1. create a stack
2. get next word f, start with word f_0
3. for next word f_i, place on stack the highest-scoring English equivalent(s) e_i, taking into account fertility and trigram history
4. repeat step 3 (i := i+1) until end of sentence
5. try replacing some e_j with a second-most likely alternative, and see how that affects the sentence score (because you also have to replace other words, etc.)

6. repeat step 5 until time runs out or until the score peaks; save the final resulting sentence
7. repeat from step 2 until end of text

This search doesn't always converge and is extremely expensive. There has been a lot of subsequent research on decoding.

Experiment of translation: Took 1000 most common English words in Hansard, got all French sentence containing just those words (1700 sentences), giving 117,000 E-F sentence pairs with *P(F|E)* 17 million parameters for the translation model. The language model had 570,000 bigrams from Hansard. Then took 73 new F sentences containing just those words, and translated them: 5% exact match; 25% very close, with alternative phrasing; 18% somewhat related in meaning; 15% not good; 37% total junk.

In DARPA evaluations (1992–94), CANDIDE topped out at about 40% (equal to most older MT systems)—essentially equal to SYSTRAN. Amazing achievement in seven years (although F-E is a very easy language pair).

Later developments in CANDIDE:

- morphology
- part of speech tagging for disambiguation
- trigram language model
- specialized recognizers for numbers, dates, names, etc.
- use Mutual Information to find fixed phrases (*Humpty Dumpty* etc.)
  $MI(w1,w2) = log\ [P(w1,w2) / (P(w1).P(w2))]$
- later tried using entropy as single standard probability count everywhere

This work, and related work using statistical (Information Theoretic) methods, had a revolutionary effect on NLP, starting in the early 1990s. The ACL conference Proceedings of 1989 and 1990, for example, had 0% papers referring to this type of work; the 1991 Proceedings started with 22%, and then on from there—a new stage in history, and a somewhat more experimental methodology.

2. Recent work

EGYPT AND REWRITE

Research on CANDIDE essentially stopped in 1995, followed by a hiatus of several years. Then Kevin Knight and colleagues at a summer school at Johns Hopkins U recreated CANDIDE in summer 1999. Build visualization engine to see alignments and fertilities. Built code to quickly and efficiently create translation models, given enough text. Developed a 100× speedup for the decoding step.

Experiment: 'Chinese MT in a day'—readable quality for about half the sentences. Much worse results with Czech (freer word order, smaller training set, lot of morphology).

EGYPT software package available free:http://www.clsp.jhu.edu/ws99/projects/mt/.

REWRITE: REWRITE project at ISI (Knight et al. 2000–). Focusing on creation of more sophisticated models, and of training on much less data. Need whole separate semester for all this.

Other Work

Several other researchers took up statistical MT and developed it in various directions. Some of the more noticeable include:

Dekai Wu, Hong Kong: Built a statistical model of tree inversion operations to start handling some syntactic phenomena, tested in English-Chinese.

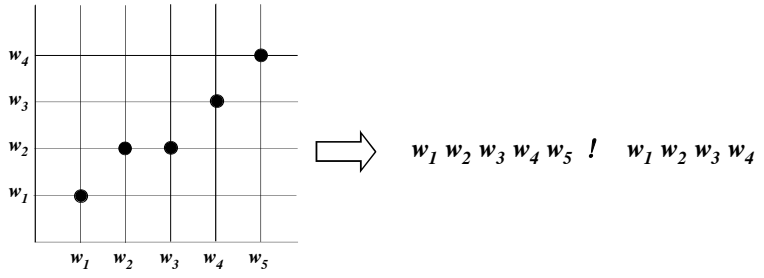
Dan Melamed, NYU: Built sophisticated algorithms to align sentences and words across parallel corpora and from this extract translation tables.

Kenji Yamada and Kevin Knight, ISI: Built a model of translation going from input sentence to syntax tree to output sentence, on the hope that one can capture the syntactic transformations within the mapping rules.

Salim Roukos, Kishore Papineni, et al., IBM New York: Worked on new models of MT using other statistical methods.

Numerous other researchers in universities all over the world tried different versions of statistical MT. The work of Philipp Koehn at the University of Edinburgh on noun phase translation is particularly interesting.

An important advance was made by Franz Och, University of Aachen, Germany, and later ISI: He built an MT system that builds not only word–word translation tables with additional fertility and distortion, but that learns ngram translation phrases, which he represents pictorially as follows:



In essence, this is an EBMT (example-based) system whose "examples" are automatically learned. Och uses Maximum Entropy to learn the best combination of feature weights to build the ngram tables. After building the massive ngram tables, the main problem is for any new sentence to be translated to select and assemble the translation ngrams into a good sentence. The system outputs a ranked list of some tens of thousands of options, and Och applies various evaluation functions to re-rank them to let the best ones appear at the top. Och's system has since around 2001 consistently been the top performer in the annual MT evaluation competitions held by DARPA. In 2004 Och left ISI to work at Google, and the current Google translation engines are his work.

3. What Makes MT Hard?

Process-related Problems

1. Lexical level

Goal: locate each source word correctly in lexicon

Problems:
- morphology: spelling errors, bad segmentation (missing or extra delimiters) "run together" or "run to get her"
- ambiguity: idioms, proper name delimitation

Tasks:
- de-inflection (morphology analysis)
- correct lexicon access

2. Syntactic level

Goal: build correct parse tree

Problems:
- extra- and non-grammaticality
- syntactic ambiguity (symbol and attachment: *time flies like an arrow*)

Tasks:
- part of speech analysis
- find appropriate grammar rule
- assemble parse tree

3. Semantic level

Goal: build correct semantic interpretation

Problems:
- ambiguity (symbol and attachment) (the man with the child vs. the man with the glasses)
- context: underspecified input
- metonymy (Washington announced that...) & metaphor

Tasks:
- find correct symbol in symbol lexicon
- find correct combination/mapping rules and constraints
- assemble semantic structure correctly

4. Discourse level

Goal: build complete interpretation of text as a whole

Problems:

- nature of discourse structure
- Speech Acts and Implicatures
- pragmatics: interlocutor goals, interpersonal effects, etc.

Tasks:
- resolve anaphora (pronouns, etc.)
- assemble discourse structure

Resource-related Problems

Core problem: the size of language!

Lexical coverage
- number of words and phrases (commercial systems: 250,000 to 1 million)
- number of proper names (easily 100,000+)

Syntactic coverage
- number of grammar rules (commercial systems: 500–1,000)

Semantic coverage
- number of internal representation symbols—more, for greater delicacy of representation and better quality: 200,000+
- number of inference rules: no-one knows

Optional further reading

Textbook: Koehn, P. 2008. *Statistical Machine Translation*. Cambridge University Press, to appear.

CANDIDE I: Brown, P.F., J. Cocke, S.A. Della Pietra, V.J. Della Pietra, F. Jelinek, J.D. Lafferty, R.L. Mercer, and P.S. Roossin. 1990. A Statistical Approach to Machine Translation. *Computational Linguistics* 16(2) (79–85).

CANDIDE II: Brown, P.F., S. Della Pietra, V. Della Pietra, R. Mercer. 1993. The Mathematics of Statistical Machine Translation: Parameter Estimation. *Computational Linguistics* 19(2) (263–311).

EGYPT: Al-Onaizan, Y., Curin, J., Jahr, M., Knight, K., Lafferty, J., Melamed, D., Och, F.-J., Purdy, D., Smith, N. A., and Yarowsky, D. 1999. Statistical Machine Translation, Final Report, JHU Workshop 1999. Technical Report, CLSP/JHU. http://www.clsp.jhu.edu/ws99/projects/mt/

REWRITE: Al-Onaizan, Y., Germann, U., Hermjakob, U., Knight, K., Koehn, P., Marcu, D., Yamada, K. 2000. Translating with Scarce Resources. *Proceedings of the American Association for Artificial Intelligence conference* (AAAI).

REWRITE: Koehn, P., and Knight, K. 2000. Estimating Word Translation Probabilities from Unrelated Monolingual Corpora using the EM Algorithm. *Proceedings of the American Association for Artificial Intelligence conference* (AAAI).

Various aspects of statistical MT: see

- Koehn: http://www.iccs.informatics.ed.ac.uk/~pkoehn/
- Och: http://www.fjoch.com/
- Knight: http://www.ii.edu/~knight