Introduction to Statistical Machine Translation

Salim Roukos
roukos@us.ibm.com

Thanks to:
Hany Hasan, Kevin Knight and Philipp Koehn

© 2010 IBM Corporation
Outline

- Introduction
- Word alignment
  - Model 1, 2, and 3
  - HMM
  - Maximum entropy model
- Machine Translation
- Phrase-based system
- Direct translation model
Machine Translation in History
The early hopes:

- 1933: Patent for a word translation & printing machine
- 1946: MT on ENIAC (Weaver et al)
- 1946-1947: Weaver (et al) realized how complex MT is.
- 1949: Weaver Memorandum (what it would take for MT)
ENIAC, 1946
Recognizing fully, even though necessarily vaguely, the semantic difficulties because of multiple meanings, etc., I have wondered if it were unthinkable to design a computer which would translate. Even if it would translate only scientific material (where the semantic difficulties are very notably less), and even if it did produce an inelegant (but intelligible) result, it would seem to me worth while... Also knowing nothing official about, but having guessed and inferred considerable about, powerful new mechanized methods in cryptography... one naturally wonders if the problem of translation could conceivably be treated as a problem in cryptography. When I look at an article in Russian, I say: “This is really written in English, but it has been coded in some strange symbols. I will now proceed to decode.”.
The early hopes: end

- 1952 – MIT Conference on MT (first small scale E-F, F-E)
- 1964 – ALPAC Report:
  - “there is no immediate or predictable prospect of useful machine translation”
  - “no need for further investment in MT research”
- 1976-1989: Systran, Logos and others developed transfer based systems.
- Till 1989: the rule-based approach dominated so far.
The rise of SMT

- 1989: IBM introduces SMT
  - *Inspired by* Weaver Memorandum
  - *Corpus-based approaches (Canadian Parliament)*
  - Empiricism vs rationalism
- 1993-1999: Few activities due to lack of open source tools
- 1999: JHU Workshop implemented open source tools for IBM SMT model
- 2000: till now: The rise of SMT as we know today
  - IBM, Language Weaver, Google Translator, Microsoft Translator
  - all are SMT systems with tens of languages
MT so far

- There is no unified approach for MT yet
- SMT is dominating the NLP field now
- This does not mean it is the best approach for MT
  - But, it is the most efficient approach so far
- SMT and sophisticated linguistics knowledge are converging
Natural Language Processing

Lexical

Morphological

Syntactic

Semantic

Natural Language Interface
- command & query translation
- output translation

Automated Translation
- bilingual dictionaries
- idiom identification
- context analysis

Speech Processing
- recognition
- generation

On-line Books
- summarization
- intelligent search
- indexing
- synonyms

Text Retrieval
- word forms
- decompounding
- synonyms

Word Processing
- synonyms
- hyphenation
- spell check
- definitions

Handwriting Recognition/OCR
- post editing

Smart Mail
- routing
- summarization
- prioritization
- indexing
MT Triangle
Full Analysis and Generation
Statistical Approach

![Diagram showing the Statistical Approach framework](image)
Practical combination
Peace with the palestinian

Peace with the palestinian
What makes MT so hard?

- Natural Languages are highly complex
- Many words have different translations
- Grammatical and lexical structures differ from language to another
- Context dependent
- Domain dependent
- Non-linguistics features: i.e. World knowledge
What is needed to perform MT

- Morphological dependencies
- Syntactic dependencies
- Semantic dependencies
- Pragmatic dependencies

Weak and vague dependencies
Rarely possible to describe simple and relevant rules
MT Approaches

- **Knowledge Based - Rule Based approach**
  - Human experts specify rules
  - Very expensive and time consuming
  - Less adaptive

- **Empirical (Data Driven) approach**
  - Knowledge automatically obtained from example translation, a parallel corpus
  - New systems could be developed very quickly
Empirical (Data Driven) Approach

- Example Based MT
  - Sentence is translated by analyzing similar previously seen translation examples.
  - Less general
  - Very large search space
- Statistical MT (SMT)
  - Translation examples are used to train a statistical translation model
  - General Approach
  - Adaptive Approach
Empirical (Data Driven) Approach
Statistical Machine Translation

- Machine learning techniques
- Statistical based approach
- Completely language independent
- Novel approaches
- Cost Effective
- Efficient Language-Independent analysis
Why Corpus-Based MT?

- the (relative) failure of rule-based approaches
- the increasing availability of machine-readable text
- the increase in capability of hardware (CPU, memory, disk space) with decrease in cost
وأعلن وزير الخارجية التركي عبد الله جول

said Gul Abdullah Minister Foreign Turkish
IBM Models and Word Alignment
SMT: Source-Channel Model

\[ \hat{e} = \text{argmax}_e P(e|f) \]

\[ P(e|f) = \frac{P(f|e)P(e)}{P(f)} \]
Source Channel Model

\[
P(e | f) = \frac{P(f | e)P(e)}{P(f)} \\
e = \arg \max_e P(f | e)P(e)
\]

Source-channel models how f speakers produce f sentences:

- They pick an English sentence \( e \sim P(e) \)
- They produce a French sentence \( F \) using \( P(f | e) \)
- Your job is to guess which sentence \( e \) they picked.

\( \hat{e} = \arg \max_e P(f | e)P(e) \)
Language Modelling

- A language model assigns a probability to every string in that language.
- A language model can be:
  - Word-based Language Model (Lexical)
  - Syntactic-based Language Model (Syntax)
- More on Language Modelling later.
Two components:

- Translation model
  \[ P(f|e) \]

- Language model
  \[ P(e) \]
The Translation Model

**Word re-ordering in translation:**

The language model establishes the probabilities of the possible orderings of a given bag of words, e.g.

\{have,programming,a,seen,never,I,language,better\}.

Effectively, the language model worries about word order, so that the translation model doesn’t have to... But what about a bag of words such as

\{loves,John,Mary\}?

Maybe the translation model *does* need to know a little about word order, after all...
Alignments

Alignment is Mapping a source word at position j to a target word at position i with a function $a_j = i$

$$f$$

$$e$$

The big house is far

$a:\{1\rightarrow 3, 2\rightarrow 2, 3\rightarrow 5\}$

$$t = p(f_j | e_{a_j})$$
Alignments

\[ f = f_1 f_2 \cdots f_m \]
\[ e = e_1 e_2 \cdots e_l \]

We need \( P(f|e) \) => introduce word alignment produce each \( f \)-word form an \( e \)-word. Which one:

For \( f \)-word \( f_j \) assume it is produce/aligned to \( e \)-word \( e_{a_j} \).
Hidden Alignment

\[ P(f|e) = \sum_a P(f, a|e) \]
IBM Model 1

- Generative model: break up translation process into smaller steps
- IBM Model 1 only uses lexical translation
- Translation probability
  - for a foreign sentence $f = f_1 f_2 \ldots f_m$
  - from an English sentence $e = e_1 e_2 \ldots e_l$
  - with an alignment of each f-word to an e-word

$$P(f, a | e) = \frac{\epsilon}{(l+1)^m} \prod_{j=1}^{m} t(f_j | e_{a_j})$$
Learning Model-1 parameters

- We would like to estimate the lexical translation probabilities \( t(e \mid f) \)
  from a parallel corpus
  - but we do not have the alignments
- Chicken and egg problem
  - if we had the alignments,
    - we could estimate the parameters of our generative model
  - if we had the parameters,
    - we could estimate the alignments
Hidden Alignment

\[ P(f|e) = \sum_a P(f, a|e) \]
Initially all connections are equally likely.

Model learns gradually that *house* is often translated as 

البيت
EM 2

- After first iteration:

Model learns that house is likely translated as البيت
EM 3

- After few iteration:

  Model learns the correct translation (converges)
EM fo IBM Model 1

- EM Algorithm consists of two steps
  - Expectation-Step: Apply model to the data
    - parts of the model are hidden (here: alignments)
    - using the model, assign probabilities to possible values
  - Maximization-Step: Estimate model from data
    - take assign values as fact
    - collect counts (weighted by probabilities)
    - estimate model from counts
- Iterate these steps until convergence
EM

E step: partial counts

\[ c(f|e; [e, f]) = \frac{t(f|e)}{t(f|e_1) + t(f|e_2) + \cdots + t(f|e_l)} \cdot occ(f) \cdot occ(e) \]

M step

\[ t(f|e) = \frac{1}{\alpha} \sum c(f|e; e_k, f_k) \]
IBM Models

- IBM Model 1: lexical translation
- IBM Model 2: adds absolute reordering model
- IBM Model 3: adds fertility model
- IBM Model 4: relative reordering model
- IBM Model 5: fixes deficiency

HMM Model:
- Words do not move independently of each other
  - they often move in groups
  - condition word movements on previous word
- HMM alignment model:
  - EM algorithm application harder, requires dynamic programming
  - IBM Model 4 is similar, also conditions on word classes
M1: Word-to-word translation

M2: Distortion model \( p(a_j | j, m, l) \)

M3: Fertility \( p(n | e) \)

Model 4 and 5

HMM jump depends on previous e-word

\[
p(f, a | e) = \pi_{j=1}^{m} p(a_j | a_{j-1})p(f_j | e_{a_j})
\]
MaxEnt alignment model

\[ p(a_j | a_{1}^{j-1}, S, T) \]
Phrase-based SMT
Phrasal Alignments in SMT

- Everything we’ve looked at so far assumes a set of word alignments.
- As speakers of foreign languages, we know that words don’t map one-to-one.
- It’d be better if we could map ‘phrases’, or sequences of words, and if need be probabilistically reorder them in translation …
- Many-to-many mappings can handle non-compositional phrases
- Local context is very useful for disambiguation:
  - Interest in \( \rightarrow \) …
  - Interest rate \( \rightarrow \) …
- The more data, the longer the learned phrases (whole sentences, sometimes …)
How to learn Phrasal Alignments

- We can learn as many phrase-to-phrase alignments as are consistent with the word alignments.
- EM training and relative frequency can give us our phrase-pair probabilities.
- We can use word alignments to get phrasal alignments.
- One alternative is the joint phrase model.
- This is called:
  - Phrase-based SMT.
Learning Phrasal Alignments

<table>
<thead>
<tr>
<th>impossible d’extraire une</th>
<th>liste ordonnée</th>
<th>des</th>
<th>services</th>
</tr>
</thead>
<tbody>
<tr>
<td>could</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>not</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>get</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>an</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ordered</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>list</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>of</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>services</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Here’s a set of English→French Word Alignments
Learning Phrasal Alignments

<table>
<thead>
<tr>
<th>impossible d’extraire une</th>
<th>liste ordonnée</th>
<th>des services</th>
</tr>
</thead>
<tbody>
<tr>
<td>could</td>
<td></td>
<td></td>
</tr>
<tr>
<td>not</td>
<td></td>
<td></td>
</tr>
<tr>
<td>get</td>
<td></td>
<td></td>
</tr>
<tr>
<td>an</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ordered</td>
<td></td>
<td></td>
</tr>
<tr>
<td>list</td>
<td></td>
<td></td>
</tr>
<tr>
<td>of</td>
<td></td>
<td></td>
</tr>
<tr>
<td>services</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Here’s a set of French→English Word Alignments
Learning Phrasal Alignments

<table>
<thead>
<tr>
<th>impossible d’extraire une</th>
<th>liste ordonnée</th>
<th>des</th>
<th>services</th>
</tr>
</thead>
<tbody>
<tr>
<td>could</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>not</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>get</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>an</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ordered</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>list</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>of</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>services</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

We can take the Intersection of both sets of Word Alignments
Learning Phrasal Alignments

<table>
<thead>
<tr>
<th>impossible d’extraire une</th>
<th>liste ordonnée</th>
<th>des</th>
<th>services</th>
</tr>
</thead>
<tbody>
<tr>
<td>could</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>not</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>get</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>an</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ordered</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>list</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>of</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>services</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Taking contiguous blocks from the Intersection gives sets of highly confident phrasal Alignments
Learning Phrasal Alignments

And back off to the Union of both sets of Word Alignments
Learning Phrasal Alignments

We can also group together contiguous blocks from the Union to give us (less confident) sets of phrasal alignments.
Learning Phrasal Alignments

<table>
<thead>
<tr>
<th>impossible d’extraire une</th>
<th>liste ordonnée</th>
<th>des services</th>
</tr>
</thead>
<tbody>
<tr>
<td>could</td>
<td></td>
<td></td>
</tr>
<tr>
<td>not</td>
<td></td>
<td></td>
</tr>
<tr>
<td>get</td>
<td></td>
<td></td>
</tr>
<tr>
<td>an</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ordered</td>
<td></td>
<td></td>
</tr>
<tr>
<td>list</td>
<td></td>
<td></td>
</tr>
<tr>
<td>of</td>
<td></td>
<td></td>
</tr>
<tr>
<td>services</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

We can also group together contiguous blocks from the Union to give us (less confident) sets of phrasal alignments.
Learning Phrasal Alignments

We can also group together contiguous blocks from the Union to give us (less confident) sets of phrasal alignments.
Learning Phrasal Alignments

We can also group together contiguous blocks from the Union to give us (less confident) sets of phrasal alignments.
Learning Phrasal Alignments

<table>
<thead>
<tr>
<th>impossible d’extraire une</th>
<th>liste ordonnée</th>
<th>des services</th>
</tr>
</thead>
<tbody>
<tr>
<td>could</td>
<td></td>
<td></td>
</tr>
<tr>
<td>not</td>
<td></td>
<td></td>
</tr>
<tr>
<td>get</td>
<td></td>
<td></td>
</tr>
<tr>
<td>an</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ordered</td>
<td></td>
<td></td>
</tr>
<tr>
<td>list</td>
<td></td>
<td></td>
</tr>
<tr>
<td>of</td>
<td></td>
<td></td>
</tr>
<tr>
<td>services</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

We can also group together contiguous blocks from the Union to give us (less confident) sets of phrasal alignments.
Decoding techniques for Statistical Machine Translation
**Decoding**

Given input string \( s \), choose the target string \( t \) that maximises \( P(t|s) \)

\[
\text{argmax } P(t|s) = \text{argmax } ( P(t) \times P(s|t) )
\]

- **Language Model**
- **Translation Model**

**Decoding Process:**
- Substitute each word/phrase by possible translation
- Build translation hypothesis graph step by step
- Score the resulting paths:
  - using the translation model and the language model
Decoding

- **Monotonic version:**
  - Substitute phrase by phrase, left to right
  - Word order can change within phrases, but phrases themselves don’t change order
  - Allows a dynamic programming solution (beam search)

- **Non-monotonic version:**
  - Explore reordering of phrases themselves
  - More complicated decoding
  - Larger search space
  - Requires more sophisticated pruning techniques
Monotonic Decoding (No Re-ordering)

- Limited capability with no re-ordering
- Very fast decoding
Monotonic Decoding (No Re-ordering)

- Limited capability with no re-ordering
- Very fast decoding
Decoding Process
Non-monotonic (with re-ordering)

• Build translation left to right
  • Select foreign words to be translated
Decoding Process

- Build translation left to right
  - Select foreign words to be translated
  - Find English phrase translation
  - Add English phrase to end of partial translation
  - Mark words as translated
Decoding Process

- One to many translation
- Re-ordering
Translation Options

• Look up possible phrase translations
  • Many different ways to segment words into phrases
  • Many different ways to translate each phrase
Hypothesis Expansion

- until all foreign words covered
- find best hypothesis that covers all foreign words
- backtrack to read off translation
Hypothesis Expansion:

• Adding more hypothesis leads to the explosion of the search space
• Number of hypotheses is exponential with respect to sentence length
• Decoding is NP-complete
• Need to reduce search space
  • risk free: hypothesis recombination
  • risky: histogram/threshold pruning
Hypothesis Recombination

Different paths to the same partial translation
- Combine paths
- drop weaker path
- keep pointer from weaker path (for lattice generation)

Thanks to Philipp Koehn
Hypothesis Pruning

- Heuristically discard weak hypotheses early
- Organize Hypothesis in stacks, e.g. by
  - same foreign words covered
  - same number of foreign words covered
- Compare hypotheses in stacks, discard bad ones
  - histogram pruning: keep top n hypotheses in each stack (e.g., n=100)
  - threshold pruning: keep hypotheses that are at most $\alpha$ times the cost of best hypothesis in stack (e.g., $\alpha=0.001$)
Phrase-Based Translation

Scoring: Try to use phrase pairs that have been frequently observed. Try to output a sentence with frequent English word sequences.

Table 1: #11# the seven - member crew includes astronauts from france and russia.
Outline

- Decoding Techniques
- Re-ordering Techniques
- Log-linear models
Phrase-based SMT
Log-Linear Model

- IBM Models deploys three components:
  - Translation model, Language Model and Distortion model

- This can be represented as weighted components:

  \[ P_{tm} * P_{lm} * P_{dist} \]

- Motivated by the need to add new components:

  \[ P_{tm}^{\lambda_1} * P_{lm}^{\lambda_2} * P_{dist}^{\lambda_3} \]

\[ \log \prod_i P_i = \sum_i \lambda_i \log P_i \]
Log-Linear model components /features

- Many different knowledge sources useful
  - Phrase translation model
  - Word translation model
  - Reordering (distortion) model
  - Word drop feature
  - Language models
  - Additional linguistics features (i.e. POS)
  - Any feature you can think could be useful
State of-the-art Features

- Source-Target phrase translation
- Target-Source phrase translation
- Source-Target word translation
- Target-Source word translation
- Distortion model
- N-gram Language Model
- Word/phrase deletion penalty
Toolkit
Introduction to MOSES

- Moses is a statistical machine translation system that allows you to automatically train translation models for any language pair.

- All you need is a collection of translated texts (parallel corpus).

- An efficient search algorithm finds quickly the highest probability translation among the exponential number of choices.
Basic Components

Translation Model (TM)

Language Model (LM)

Decoder

Preprocessing

Post-processing

Source

Target

Input

Translation
Basic Components – Used Toolkits

- Language Model: SRILM Toolkit

- Translation Model
  - GIZA ++ Toolkit for word alignments
  - Heuristics to build phrase table

- Decoder: Stack decoding algorithm
  - Requires:
    - Phrase Table: Phrase Translation table
    - Moses.ini: The configuration file for the decoder
    - Language Model File
Future topics

- Syntax-based models
  - Source syntax
  - Target syntax
  - Tree-to-Tree models

- Factored models $p(f|e)$
Thankyou