Novel Inference, Training and Decoding Methods over Translation Forests

Zhifei Li
Center for Language and Speech Processing
Computer Science Department
Johns Hopkins University

Advisor: Sanjeev Khudanpur
Co-advisor: Jason Eisner
Statistical Machine Translation Pipeline
Statistical Machine Translation Pipeline

Bilingual Data
Statistical Machine Translation Pipeline

Bilingual Data → Generative Training → Translation Models
Statistical Machine Translation Pipeline

Bilingual Data ➔ Generative Training ➔ Translation Models

Monolingual English
Statistical Machine Translation Pipeline

Bilingual Data → Generative Training → Translation Models

Monolingual English → Generative Training → Language Models
Statistical Machine Translation Pipeline

Bilingual Data

Generative Training

Translation Models

Held-out Bilingual Data

Discriminative Training

Optimal Weights

Monolingual English

Generative Training

Language Models
Statistical Machine Translation Pipeline

Bilingual Data → Generative Training → Translation Models

Monolingual English → Generative Training → Language Models

Held-out Bilingual Data → Discriminative Training → Optimal Weights
Statistical Machine Translation Pipeline

- Bilingual Data
  - Generative Training
  - Translation Models
  - Held-out Bilingual Data
  - Discriminative Training
  - Optimal Weights

- Monolingual English
  - Generative Training
  - Language Models

- Unseen Sentences
Statistical Machine Translation Pipeline

Bilingual Data

Monolingual English

Generative Training

Generative Training

Translation Models

Language Models

Discriminative Training

Held-out Bilingual Data

Optimal Weights

Unseen Sentences

Decoding

Translation Outputs
Training a Translation Model
Training a Translation Model

垫子 上 的 猫
dianzi shang de mao
Training a Translation Model

dianzi shang de mao

a cat on the mat
Training a Translation Model

垫子 上 的 猫
dianzi shang de mao

a cat on the mat
Training a Translation Model

![Image of a cat on a mat]

Training a Translation Model

垫子 上 的 猫

dianzi shang de mao

a cat on the mat

\[ X \rightarrow \langle \text{dianzi shang, the mat} \rangle \]
1.2.1 Translation-Sense Ambiguity

As in many other natural language processing (NLP) problems, ambiguity is a central issue in machine translation. Broadly speaking, there are two kinds of ambiguities in translation: (1) literal meaning ambiguity, and (2) word sense ambiguity. Here, we focus on the latter.

In a natural language, the same word may have different senses-meanings, depending on the context. For example, the word “bank” can be referring to a financial bank or the edge of a river. This kind of ambiguity is called translation-sense ambiguity.

Example of decoding a test sentence.

For the test sentence “垫子上的猫,” which means “a cat on the mat,” and its English translation “the dog on the mat,” we have different meaning in Chinese. Specifically, the first rule has the Chinese sentence “垫子上的猫,” which means “a cat on the mat,” and the second rule has the Chinese sentence “猫在床上,” which means “the cat is on the bed.”

As we can see, there are two different translations for the same sentence. To generate a derivation tree in English, we need to have the rules in the test grammar to avoid spurious ambiguity.

6a. Hiero rules in the test grammar

6b. derivation tree whose English yield is “the dog on the mat”
Training a Translation Model

垫子 上 的 猫
dianzi shang de

X → ⟨ dianzi shang, the mat ⟩
X → ⟨ mao, a cat ⟩
Training a Translation Model

Example of decoding a test sentence.

\[ X \rightarrow \langle \text{dianzi shang, the mat} \rangle \]
\[ X \rightarrow \langle \text{mao, a cat} \rangle \]
Training a Translation Model

\[
X \rightarrow \langle \text{dianzi shang}, \text{the mat} \rangle \\
X \rightarrow \langle \text{mao}, \text{a cat} \rangle \\
X \rightarrow \langle \text{dianzi shang de } X_0, X_0 \text{ on the mat} \rangle
\]
Training a Translation Model

垫子 上 的 猫

\[ X_0 \text{ de mao} \]

\[ X_0 \text{ on the mat} \]

\[ X \rightarrow \langle \text{dianzi shang, the mat} \rangle \]

\[ X \rightarrow \langle \text{mao, a cat} \rangle \]

\[ X \rightarrow \langle \text{dianzi shang de } X_0, X_0 \text{ on the mat} \rangle \]
Training a Translation Model

\[
\begin{align*}
X & \rightarrow \langle \text{dianzi shang}, \text{the mat} \rangle \\
X & \rightarrow \langle \text{mao}, \text{a cat} \rangle \\
X & \rightarrow \langle \text{dianzi shang de } X_0, X_0 \text{ on the mat} \rangle \\
X & \rightarrow \langle X_0 \text{ de mao}, \text{a cat on } X_0 \rangle
\end{align*}
\]
Training a Translation Model

Two example Hiero rules for Chinese-to-English translation are:

1. \( X \rightarrow \langle \text{dianzi shang}, \text{the mat} \rangle \)
2. \( X \rightarrow \langle \text{mao}, \text{a cat} \rangle \)
3. \( X \rightarrow \langle \text{dianzi shang de} \ X_0, \ X_0 \text{ on the mat} \rangle \)
4. \( X \rightarrow \langle X_0 \text{ de mao}, \text{a cat on} \ X_0 \rangle \)
Training a Translation Model

垫子 上 的 猫

\[
X_0 \rightarrow \langle \text{dianzi shang} \text{, the mat} \rangle
\]

\[
X \rightarrow \langle \text{mao} \text{, a cat} \rangle
\]

\[
X \rightarrow \langle \text{dianzi shang de } X_0 \text{, } X_0 \text{ on the mat} \rangle
\]

\[
X \rightarrow \langle X_0 \text{ de } \text{mao} \text{, a cat on } X_0 \rangle
\]

\[
X \rightarrow \langle X_0 \text{ de } X_1 \text{, } X_1 \text{ on } X_0 \rangle
\]
Decoding a Test Sentence
Decoding a Test Sentence
Decoding a Test Sentence

垫子 上 的 狗
Decoding a Test Sentence

垫子  上 的 狗
dianzi  shang de gou
Decoding a Test Sentence

垫子 上 的 狗
dianzi shang de gou
the dog on the mat
Decoding a Test Sentence

垫子上 的 狗

dianzi shang de gou

the dog on the mat

\[
\begin{align*}
X & \rightarrow \langle \text{dianzi shang, the mat} \rangle \\
X & \rightarrow \langle \text{gou, the dog} \rangle \\
X & \rightarrow \langle X_0 \text{ de } X_1, X_1 \text{ on } X_0 \rangle \\
S & \rightarrow \langle X_0, X_0 \rangle
\end{align*}
\]
Decoding a Test Sentence

垫子 上 的 狗

dianzi shang de gou
dog on the mat

X → ⟨ dianzi shang, the mat ⟩
X → ⟨ gou, the dog ⟩
X → ⟨ X₀ de X₁, X₁ on X₀ ⟩
S → ⟨ X₀, X₀ ⟩
Decoding a Test Sentence

垫子 上 的 狗
dianzi shang de gou
the dog on the mat

\[
\begin{align*}
X & \rightarrow \langle \text{dianzi shang, the mat} \rangle \\
\text{X} & \rightarrow \langle \text{gou, the dog} \rangle \\
X & \rightarrow \langle X_0 \text{ de } X_1, X_1 \text{ on } X_0 \rangle \\
S & \rightarrow \langle X_0, X_0 \rangle
\end{align*}
\]
Decoding a Test Sentence

垫子 上 的 狗
dianzi shang de gou
the dog on the mat

\[
X \rightarrow \langle \text{dianzi shang}, \text{the mat} \rangle
\]

\[
X \rightarrow \langle \text{gou}, \text{the dog} \rangle
\]

\[
X \rightarrow \langle X_0 \text{ de } X_1, X_1 \text{ on } X_0 \rangle
\]

\[
S \rightarrow \langle X_0, X_0 \rangle
\]

dianzi shang de gou
Decoding a Test Sentence

垫子 上 的 狗
dianzi shang de gou
the dog on the mat

\[
X \rightarrow \langle \text{dianzi shang, the mat} \rangle
\]

\[
X \rightarrow \langle \text{gou, the dog} \rangle
\]

\[
X \rightarrow \langle X_0 \text{ de } X_1, X_1 \text{ on } X_0 \rangle
\]

\[
S \rightarrow \langle X_0, X_0 \rangle
\]
Decoding a Test Sentence

垫子上 的 狗
dianzi shang de gou
the dog on the mat

\[
\begin{align*}
X & \rightarrow \langle \text{dianzi shang, the mat} \rangle \\
X & \rightarrow \langle \text{gou, the dog} \rangle \\
X & \rightarrow \langle X_0 \text{ de } X_1, X_1 \text{ on } X_0 \rangle \\
S & \rightarrow \langle X_0, X_0 \rangle
\end{align*}
\]

Decoding a Test Sentence

\[
\begin{align*}
X & \rightarrow \langle \text{dianzi shang, the mat} \rangle \\
| \\
\text{dianzi shang de gou} \\
| \\
X & \rightarrow \langle \text{gou, the dog} \rangle
\end{align*}
\]

1.2 Ambiguity in Language Translation

As in many other natural language processing (NLP) problems, ambiguity is a central issue in machine translation. Broadly speaking, there are two kinds of ambiguities in translation:

1.2.1 Translation-Sense Ambiguity

In a natural language, the same word may have different senses-meanings, depending on the context. For example, the word “bank” can be referring to a financial bank or the edge of a river. This kind of ambiguity is called word sense ambiguity. Clearly, when we translate such an ambiguous word, different translations should be used depending on the context. For example, a translation model for a translation task from English to Chinese, may contain two rules as follows:

\[
\begin{align*}
X & \rightarrow \langle \text{bank, he an} \rangle \\
X & \rightarrow \langle \text{bank, yin hang} \rangle
\end{align*}
\]

where they have the same English side (i.e., “bank”), but have different Chinese sides, which have different meaning in Chinese. Specifically, the first rule has the Chinese “he an” which means the edge of a river, while the second one has the Chinese “yin hang” which means a
Decoding a Test Sentence

X → ⟨ dianzi shang, the mat ⟩
X → ⟨ gou, the dog ⟩
X → ⟨ X₀ de X₁, X₁ on X₀ ⟩
S → ⟨ X₀, X₀ ⟩

1.2 Ambiguity in Language Translation

As in many other natural language processing problems, ambiguity is a central issue in machine translation. Broadly speaking, there are two kinds of ambiguities in translation: translation-sense ambiguity and spurious ambiguity.

1.2.1 Translation-Sense Ambiguity
In a natural language, the same word may have different senses—meanings—depending on the context. For example, the word "bank" can be referring to a financial bank or the edge of a river. This kind of ambiguity is called *word sense ambiguity*. Clearly, when we translate such an ambiguous word, different translations should be used depending on the context. For example, a translation model for a translation task from English to Chinese may contain two rules as follows:

X → ⟨ bank, he an ⟩
X → ⟨ bank, yin hang ⟩

where they have the same English side (i.e., "bank"), but have different Chinese sides, which have different meaning in Chinese. Specifically, the first rule has the Chinese "he an" which means the edge of a river, while the second one has the Chinese "yin hang" which means a
Decoding a Test Sentence

垫子上的狗

The dog on the mat

dianzi shang de gou

the dog on the mat

\[
X \rightarrow \langle \text{dianzi shang, the mat} \rangle \\
X \rightarrow \langle \text{gou, the dog} \rangle \\
X \rightarrow \langle X_0 \ de \ X_1, X_1 \ on \ X_0 \rangle \\
S \rightarrow \langle X_0, X_0 \rangle
\]

\[
S \rightarrow \langle X_0, X_0 \rangle \\
X \rightarrow \langle X_0 \ de \ X_1, X_1 \ on \ X_0 \rangle \\
X \rightarrow \langle \text{dianzi shang, the mat} \rangle \\
| \ \\
| \\
dianzi shang de gou \\
| \ \\
| \\
| \\
| \\
gou, the dog
\]
Decoding a Test Sentence

垫子上的狗
the dog on the mat

dianzi shang de gou

Derivation Tree

\[
\begin{align*}
S & \rightarrow \langle X_0, X_0 \rangle \\
X & \rightarrow \langle X_0 \text{ de } X_1, X_1 \text{ on } X_0 \rangle \\
X & \rightarrow \langle \text{dianzi shang, the mat} \rangle \\
X & \rightarrow \langle \text{gou, the dog} \rangle \\
dianzi shang \text{ de } gou
\end{align*}
\]
Decoding a Test Sentence

垫子 上 的 狗

dianzi shang de gou

the dog on the mat

\[
\begin{align*}
X & \rightarrow \langle \text{dianzi shang}, \text{the mat} \rangle \\
X & \rightarrow \langle \text{gou}, \text{the dog} \rangle \\
X & \rightarrow \langle X_0 \text{ de } X_1, X_1 \text{ on } X_0 \rangle \\
S & \rightarrow \langle X_0, X_0 \rangle
\end{align*}
\]

Derivation Tree

Translation is easy?
Translation Ambiguity

dianzi shang de mao

垫子上的猫

a cat on the mat
Translation Ambiguity

垫子上的猫
dianzi shang de mao

a cat on the mat

X→⟨X₀ de X₁, X₁ on X₀⟩
Translation Ambiguity

垫子 上 的 猫
dianzi shang de mao
da cat on the mat

X→⟨X₀ de X₁, X₁ on X₀⟩

zhongguo de shoudu
capital of China
Translation Ambiguity

垫子上的猫

A cat on the mat

zhongguo de shoudu

Capital of China

X₀ → (X₀ de X₁, X₁ on X₀)
Translation Ambiguity

垫子上的猫
dianzi shang de mao

a cat on the mat

zhongguo de shoudu

capital of China

X→⟨X₀ de X₁, X₁ on X₀⟩

X→⟨X₀ de X₁, X₁ of X₀⟩
Translation Ambiguity

垫子上的猫
dianzi shang de mao

a cat on the mat

X→⟨X₀ de X₁, X₁ on X₀⟩

zhongguo de shoudu
capital of China

wo de mao
my cat

X→⟨X₀ de X₁, X₁ of X₀⟩
<table>
<thead>
<tr>
<th>Translation</th>
<th>Chinese Characters</th>
<th>English Translation</th>
</tr>
</thead>
<tbody>
<tr>
<td>dianzi shang de mao</td>
<td>垫子上的猫</td>
<td>a cat on the mat</td>
</tr>
<tr>
<td>zhongguo de shoudu</td>
<td>中国的首都</td>
<td>capital of China</td>
</tr>
<tr>
<td>wo de mao</td>
<td>我的猫</td>
<td>my cat</td>
</tr>
</tbody>
</table>

Translation Ambiguity

- \( X \rightarrow \langle X_0 \ de \ X_1, X_1 \ on \ X_0 \rangle \)
- \( X \rightarrow \langle X_0 \ de \ X_1, X_1 \ of \ X_0 \rangle \)
- \( X \rightarrow \langle X_0 \ de \ X_1, X_0 \ X_1 \rangle \)
Translation Ambiguity

垫子上的猫
a cat on the mat

zhongguo de shoudu
capital of China

wo de mao
my cat
zhifei de mao
zhifei’s cat

X→⟨X₀ de X₁, X₁ on X₀⟩

X→⟨X₀ de X₁, X₁ of X₀⟩

X→⟨X₀ de X₁, X₀ X₁⟩
Translation Ambiguity

垫子上的猫
a cat on the mat

X→⟨X₀ de X₁, X₁ on X₀⟩

zhongguo de shoudu
capital of China

X→⟨X₀ de X₁, X₁ of X₀⟩

wo de mao
my cat

X→⟨X₀ de X₁, X₀ X₁⟩

zhifei de mao
zhifei's cat

X→⟨X₀ de X₁, X₀ 's X₁⟩
电子上的猫
X → mao, a cat
X → dianzi shang, the mat
S → X₀, X₀
X → X₀ de X₁, X₁ of X₀

dianzi shang de mao
dianzi shang de mao

Joshua
(chart parser)

X → mao, a cat
X → dianzi shang, the mat
S → X_0, X_0

X → X_0 de X_1, X_1 on X_0

X → X_0 de X_1, X_1 of X_0

X → mao, a cat
X → dianzi shang, the mat
S → X_0, X_0
Joshua (chart parser)
dianzi shang de mao
Joshua (chart parser)
A hypergraph encodes four different derivation trees as shown in the four figures below. Rectangles represent items (or nodes), where each item is identified by the non-terminal symbol and source span. An item has one or more incoming hyperedges, which represent different ways of deriving the item. A hyperedge consists of a rule, and a pointer to an antecedent item for each non-terminal symbol in the rule.

Translation:
(a) A cat on the mat
(b) The mat a cat
(c) A cat of the mat
(d) The mat's a cat

Figure 1.6: A toy hypergraph generated for the Chinese input "dianzi shang de mao," and the four derivation trees the hypergraph encodes.

(Please refer to the actual page for the full diagram and text.)
(a) A hypergraph encodes four different derivation trees as shown in the four figures below. Rectangles represent items (or nodes), where each item is identified by the non-terminal symbol and source span. An item has one or more incoming hyperedges, which represent different ways of deriving the item. A hyperedge consists of a rule, and a pointer to an antecedent item for each non-terminal symbol in the rule.

(b) Translation: a cat on the mat

(c) Translation: the mat a cat

(d) Translation: a cat of the mat

(e) Translation: the mat's a cat

Figure 1.6: A toy hypergraph generated for the Chinese input "dianzi shang de mao," and the four derivation trees the hypergraph encodes.
A hypergraph is a compact data structure to encode exponentially many trees.

Figure 1.6: A toy hypergraph generated for the Chinese input "dianzi shang de mao," and the four derivation trees the hypergraph encodes.
A hypergraph is a compact data structure to encode exponentially many trees.
A hypergraph is a compact data structure to encode exponentially many trees.

Figure 1.6: A toy hypergraph generated for the Chinese input "dianzi shang de mao," and the four derivation trees the hypergraph encodes.
A hypergraph is a compact data structure to encode exponentially many trees.

Figure 1.6: A toy hypergraph generated for the Chinese input "dianzi shang de mao," and the four derivation trees the hypergraph encodes.
A hypergraph is a compact data structure to encode exponentially many trees.
A hypergraph is a compact data structure to encode exponentially many trees.
A hypergraph is a compact data structure to encode exponentially many trees.
A hypergraph is a compact data structure to encode \textit{exponentially many} trees.
A hypergraph is a compact data structure to encode **exponentially many** trees.

(a) A hypergraph encodes four different derivation trees as shown in the four figures below. Rectangles represent items (or nodes), where each item is identified by the non-terminal symbol and source span. An item has one or more incoming hyperedges, which represent different ways of deriving the item. A hyperedge consists of a rule, and a pointer to an antecedent item for each non-terminal symbol in the rule.

(b) Translation: a cat on the mat

(c) Translation: the mat a cat

(d) Translation: a cat of the mat

(e) Translation: the mat's a cat

Figure 1.6: A toy hypergraph generated for the Chinese input "dianzi shang de mao," and the four derivation trees the hypergraph encodes.
A hypergraph is a compact data structure to encode \textbf{exponentially many trees}.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{hypergraph.png}
\caption{A toy hypergraph generated for the Chinese input "dianzi shang de mao," and the four derivation trees the hypergraph encodes.}
\end{figure}
A hypergraph is a compact data structure to encode \textit{exponentially many} trees.
A hypergraph is a compact data structure to encode exponentially many trees.
A hypergraph is a compact data structure to encode exponentially many trees.

\[
\begin{align*}
S &\rightarrow \langle X_0, X_0 \rangle \\
X &\rightarrow \langle X_0 \text{ de } X_1, X_1 \text{ on } X_0 \rangle \\
X &\rightarrow \langle X_0 \text{ de } X_1, X_0 \text{'s } X_1 \rangle \\
X &\rightarrow \langle X_0 \text{ de } X_1, X_1 \text{ of } X_0 \rangle \\
X &\rightarrow \langle \text{dianzi shang, the mat} \rangle \\
X &\rightarrow \langle \text{mao, a cat} \rangle
\end{align*}
\]
A hypergraph is a compact data structure to encode exponentially many trees.
A hypergraph is a compact data structure to encode exponentially many trees.

Figure 1.6: A toy hypergraph generated for the Chinese input "dianzi shang de mao," and the four derivation trees the hypergraph encodes.
A hypergraph is a compact data structure to encode exponentially many trees.
Why Hypergraphs?

• Contains a much larger hypothesis space than a $k$-best list

• General compact data structure
  
  • special cases include
    
    • finite state machine (e.g., lattice),
    
    • and/or graph
    
    • packed forest

• can be used for speech, parsing, tree-based MT systems, and many more
(a) A hypergraph encodes four different derivation trees as shown in the four figures below. Rectangles represent items (or nodes), where each item is identified by the non-terminal symbol and source span. An item has one or more incoming hyperedges, which represent different ways of deriving the item. A hyperedge consists of a rule, and a pointer to an antecedent item for each non-terminal symbol in the rule.

(b) Translation: a cat on the mat

c) Translation: the mat a cat

d) Translation: a cat of the mat

e) Translation: the mat's a cat

Figure 1.6: A toy hypergraph generated for the Chinese input "dianzi shang de mao," and the four derivation trees the hypergraph encodes.
A hypergraph encodes four different derivation trees as shown in the four figures below. Rectangles represent items (or nodes), where each item is identified by the non-terminal symbol and source span. An item has one or more incoming hyperedges, which represent different ways of deriving the item. A hyperedge consists of a rule, and a pointer to an antecedent item for each non-terminal symbol in the rule.

Translation:
(a) a cat on the mat
(b) the mat a cat
(c) a cat of the mat
(d) the mat's a cat

Figure 1.6: A toy hypergraph generated for the Chinese input "dianzi shang de mao," and the four derivation trees the hypergraph encodes.
A hypergraph encodes four different derivation trees as shown in the four figures below. Rectangles represent items (or nodes), where each item is identified by the non-terminal symbol and source span. An item has one or more incoming hyperedges, which represent different ways of deriving the item. A hyperedge consists of a rule, and a pointer to an antecedent item for each non-terminal symbol in the rule.

Translation:
- (a) A cat on the mat
- (b) The mat a cat
- (c) A cat of the mat
- (d) The mat's a cat

Figure 1.6: A toy hypergraph generated for the Chinese input "dianzi shang de mao."
A hypergraph encodes four different derivation trees as shown in the four figures below. Rectangles represent items (or nodes), where each item is identified by the non-terminal symbol and source span. An item has one or more incoming hyperedges, which represent different ways of deriving the item. A hyperedge consists of a rule, and a pointer to an antecedent item for each non-terminal symbol in the rule.

Translation:
- a cat on the mat
- the mat a cat
- a cat of the mat
- the mat's a cat

Figure 1.6: A toy hypergraph generated for the Chinese input "dianzi shang de mao," and the four derivation trees the hypergraph encodes.
(a) A hypergraph encodes four different derivation trees as shown in the four figures below. Rectangles represent items (or nodes), where each item is identified by the non-terminal symbol and source span. An item has one or more incoming hyperedges, which represent different ways of deriving the item. A hyperedge consists of a rule, and a pointer to an antecedent item for each non-terminal symbol in the rule.

(b) Translation: a cat on the mat

c) Translation: the mat a cat

d) Translation: a cat of the mat

e) Translation: the mat's a cat

Figure 1.6: A toy hypergraph generated for the Chinese input "dianzi shang de mao," and the four derivation trees the hypergraph encodes.
(a) A hypergraph encodes four different derivation trees as shown in the four figures below. Rectangles represent items (or nodes), where each item is identified by the non-terminal symbol and source span. An item has one or more incoming hyperedges, which represent different ways of deriving the item. A hyperedge consists of a rule, and a pointer to an antecedent item for each non-terminal symbol in the rule.

(b) Translation: a cat on the mat

(c) Translation: the mat a cat

(d) Translation: a cat of the mat

(e) Translation: the mat's a cat

Figure 1.6: A toy hypergraph generated for the Chinese input "dianzi shang de mao," and the four derivation trees the hypergraph encodes.
A hypergraph encodes four different derivation trees as shown in the four figures below. Rectangles represent items (or nodes), where each item is identified by the non-terminal symbol and source span. An item has one or more incoming hyperedges, which represent different ways of deriving the item. A hyperedge consists of a rule, and a pointer to an antecedent item for each non-terminal symbol in the rule.

Translation:
- a cat on the mat
- the mat a cat
- a cat of the mat
- the mat's a cat

Figure 1.6: A toy hypergraph generated for the Chinese input "dianzi shang de mao," and the four derivation trees the hypergraph encodes.
Log-linear model:

\[ p(d \mid x) = \frac{e^{\theta \cdot \Phi(d, x)}}{Z(x)} \]

\[ Z = 2 + 1 + 3 + 2 = 8 \]
(a) A hypergraph encodes four different derivation trees as shown in the four figures below. Rectangles represent items (or nodes), where each item is identified by the non-terminal symbol and source span. An item has one or more incoming hyperedges, which represent different ways of deriving the item. A hyperedge consists of a rule, and a pointer to an antecedent item for each non-terminal symbol in the rule.

(b) Translation:
- a cat on the mat
- the mat a cat
- a cat of the mat
- the mat's a cat

Figure 1.6: A toy hypergraph generated for the Chinese input "dianzi shang de mao," and the four derivation trees the hypergraph encodes.

Log-linear model:
\[
p(d \mid x) = \frac{e^{\theta \cdot \Phi(d, x)}}{Z(x)}
\]

\[Z = 2 + 1 + 3 + 2 = 8\]

\[p = \frac{2}{8}\]

\[p = \frac{3}{8}\]

\[p = \frac{1}{8}\]
A hypergraph encodes four different derivation trees as shown in the four figures below. Rectangles represent items (or nodes), where each item is identified by the non-terminal symbol and source span. An item has one or more incoming hyperedges, which represent different ways of deriving the item. A hyperedge consists of a rule, and a pointer to an antecedent item for each non-terminal symbol in the rule.

(a) Translation: a cat on the mat
(b) Translation: the mat a cat
(c) Translation: a cat of the mat
(d) Translation: the mat's a cat

Figure 1.6: A toy hypergraph generated for the Chinese input "dianzi shang de mao," and the four derivation trees the hypergraph encodes.
The hypergraph defines a probability distribution over trees!
The hypergraph defines a probability distribution over trees! the distribution is parameterized by $\Theta$. 

(a) A hypergraph encodes four different derivation trees as shown in the four figures below. Rectangles represent items (or nodes), where each item is identified by the non-terminal symbol and source span. An item has one or more incoming hyperedges, which represent different ways of deriving the item. A hyperedge consists of a rule, and a pointer to an antecedent item for each non-terminal symbol in the rule.

(b) Translation: 

"a cat on the mat"

(c) Translation: 

"the mat a cat"

(d) Translation: 

"a cat of the mat"

(e) Translation: 

"the mat 's a cat"

Figure 1.6: A toy hypergraph generated for the Chinese input "dianzi shang de mao," and the four derivation trees the hypergraph encodes.
A hypergraph encodes four different derivation trees as shown in the four figures below. Rectangles represent items (or nodes), where each item is identified by the non-terminal symbol and source span. An item has one or more incoming hyperedges, which represent different ways of deriving the item. A hyperedge consists of a rule, and a pointer to an antecedent item for each non-terminal symbol in the rule.

Translation:
(a) A cat on the mat
(b) The mat a cat
(c) A cat of the mat
(d) The mat's a cat

The hypergraph defines a probability distribution over trees! the distribution is parameterized by $\Theta$. 

Figure 1.6: A toy hypergraph generated for the Chinese input "dianzi shang de mao," and the four derivation trees the hypergraph encodes.
The hypergraph defines a probability distribution over trees! the distribution is parameterized by Θ

Which translation do we present to a user? Decoding
The hypergraph defines a probability distribution over trees!
the distribution is parameterized by $\Theta$

Which translation do we present to a user?  
Decoding

How do we set the parameters $\Theta$?  
Training
The hypergraph defines a probability distribution over trees! the distribution is parameterized by $\Theta$.
The hypergraph defines a probability distribution over trees! The distribution is parameterized by $\Theta$.

Which translation do we present to a user? Decoding

How do we set the parameters $\Theta$? Training

What atomic operations do we need to perform? Atomic Inference

Probabilistic Hypergraph

<table>
<thead>
<tr>
<th>Training</th>
<th>Decoding</th>
</tr>
</thead>
<tbody>
<tr>
<td>(e.g., mert)</td>
<td>(e.g., mbr)</td>
</tr>
</tbody>
</table>

**atomic inference operations**
(e.g., finding one-best, k-best or expectation, inference can be *exact* or *approximate*)

Which translation do we present to a user? Decoding

How do we set the parameters $\Theta$? Training

What atomic operations do we need to perform? Atomic Inference
A hypergraph encodes four different derivation trees as shown in the four figures below. Rectangles represent items (or nodes), where each item is identified by the non-terminal symbol and source span. An item has one or more incoming hyperedges, which represent different ways of deriving the item. A hyperedge consists of a rule, and a pointer to an antecedent item for each non-terminal symbol in the rule.

(b) Translation: a cat on the mat
(c) Translation: the mat a cat
(d) Translation: a cat of the mat
(e) Translation: the mat's a cat

Figure 1.6: A toy hypergraph generated for the Chinese input "dianzi shang de mao," and the four derivation trees the hypergraph encodes.

Probabilistic Hypergraph

<table>
<thead>
<tr>
<th>training (e.g., mert)</th>
<th>decoding (e.g., mbr)</th>
</tr>
</thead>
<tbody>
<tr>
<td>atomic inference operations</td>
<td></td>
</tr>
<tr>
<td>(e.g., finding one-best, k-best or expectation, inference can be exact or approximate)</td>
<td></td>
</tr>
</tbody>
</table>

Which translation do we present to a user? Decoding
How do we set the parameters Θ? Training
What atomic operations do we need to perform? Atomic Inference
Why are the problems difficult?
The hypergraph defines a probability distribution over trees! the distribution is parameterized by $\Theta$.

### Probabilistic Hypergraph

- **Training** (e.g., mert)
- **Decoding** (e.g., mbr)

<table>
<thead>
<tr>
<th>atomic inference operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>(e.g., finding one-best, k-best or expectation, inference can be exact or approximate)</td>
</tr>
</tbody>
</table>

### Decoding

- Which translation do we present to a user?
- How do we set the parameters $\Theta$?
- What atomic operations do we need to perform?

### Training

- Why are the problems difficult?
  - brute-force will be too slow as there are exponentially many trees, so require sophisticated dynamic programs

---

(a) A hypergraph encodes four different derivation trees as shown in the four figures below. Rectangles represent items (or nodes), where each item is identified by the non-terminal symbol and source span. An item has one or more incoming hyperedges, which represent different ways of deriving the item. A hyperedge consists of a rule, and a pointer to an antecedent item for each non-terminal symbol in the rule.

(b) Translation: a cat on the mat

(c) Translation: the mat a cat

(d) Translation: a cat of the mat

(e) Translation: the mat's a cat

Figure 1.6: A toy hypergraph generated for the Chinese input "dianzi shang de mao," and the four derivation trees the hypergraph encodes.
The hypergraph defines a probability distribution over trees! the distribution is parameterized by $\Theta$

<table>
<thead>
<tr>
<th>training (e.g., mert)</th>
<th>decoding (e.g., mbr)</th>
</tr>
</thead>
<tbody>
<tr>
<td>atomic inference operations</td>
<td></td>
</tr>
<tr>
<td>(e.g., finding one-best, k-best or expectation, inference can be exact or approximate)</td>
<td></td>
</tr>
</tbody>
</table>

Which translation do we present to a user? **Decoding**

How do we set the parameters $\Theta$? **Training**

What atomic operations do we need to perform? **Atomic Inference**

Why are the problems difficult?
- brute-force will be too slow as there are exponentially many trees, so require sophisticated dynamic programs
- sometimes intractable, require approximations
Inference, Training and Decoding on Hypergraphs

• Atomic Inference
  • finding one-best derivations

<table>
<thead>
<tr>
<th>Graph</th>
<th>Topological</th>
<th>Best-first</th>
</tr>
</thead>
<tbody>
<tr>
<td>FSA</td>
<td>Viterbi</td>
<td>Dijkstra</td>
</tr>
<tr>
<td>Hypergraph</td>
<td>CYK</td>
<td>A*</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Generalized $A^*$</td>
</tr>
</tbody>
</table>

• finding k-best derivations
• computing expectations (e.g., of features)

• Training
  • Perceptron, conditional random field (CRF), minimum error rate training (MERT), minimum risk, and MIRA

• Decoding
  • Viterbi decoding, maximum a posterior (MAP) decoding, and minimum Bayes risk (MBR) decoding
Outline

• Hypergraph as Hypothesis Space

• Unsupervised Discriminative Training
  ‣ minimum imputed risk
  ‣ contrastive language model estimation

• Variational Decoding

• First- and Second-order Expectation Semirings
Outline

• Hypergraph as Hypothesis Space

• Unsupervised Discriminative Training
  ▸ minimum imputed risk
  ▸ contrastive language model estimation

• Variational Decoding

• First- and Second-order Expectation Semirings
Outline

• Hypergraph as Hypothesis Space

• Unsupervised Discriminative Training
  ▸ minimum imputed risk
  ▸ contrastive language model estimation

main focus

• Variational Decoding

• First- and Second-order Expectation Semirings

---

Diagram:

- Bilingual Data
- Monolingual English
- Generative Training
- Translation Models
- Language Models
- Discriminative Training
- Optimal Weights
- Held-out Bilingual Data
- Unseen Sentences
- Decoding
- Translation Outputs
Training Setup

- Each **training example** consists of:
  - a foreign sentence (from which a hypergraph is generated to represent many possible translations)
  - a reference translation

  \[
  x: \text{dianzi shang de mao} \\
  y: \text{a cat on the mat}
  \]

- **Training**
  - adjust the parameters $\Theta$ so that the reference translation is preferred by the model

\[X \rightarrow \langle \text{dianzi shang, the mat} \rangle \\
X \rightarrow \langle \text{mao, a cat} \rangle \]
Supervised: Minimum Empirical Risk
Supervised: Minimum Empirical Risk

- Minimum Empirical Risk Training

\[ \theta^* = \arg \min_{\theta} \sum_{x,y} \hat{p}(x,y)L(\delta_{\theta}(x), y) \]
Supervised: Minimum Empirical Risk

- Minimum Empirical Risk Training

\[ \theta^* = \arg \min_{\theta} \sum_{x,y} \tilde{p}(x, y) L(\delta_{\theta}(x), y) \]

empirical distribution
Supervised: Minimum Empirical Risk

- **Minimum Empirical Risk Training**

\[ \theta^* = \arg \min_{\theta} \sum_{x,y} \tilde{p}(x, y) L(\delta_{\theta}(x), y) \]
Supervised: Minimum Empirical Risk

- Minimum Empirical Risk Training

\[ \theta^* = \arg \min_{\theta} \sum_{x,y} \tilde{p}(x, y) L(\delta_{\theta}(x), y) \]
Supervised: Minimum Empirical Risk

- Minimum Empirical Risk Training

\[ \theta^* = \arg \min_{\theta} \sum_{x, y} \tilde{p}(x, y) L(\delta_{\theta}(x), y) \]

empirical distribution

\( x \rightarrow \delta_{\theta} \)
Supervised: Minimum Empirical Risk

- **Minimum Empirical Risk Training**

\[
\theta^* = \arg \min_{\theta} \sum_{x,y} \tilde{p}(x, y) L(\delta_\theta(x), y)
\]

MT decoder

empirical distribution

\[x \rightarrow \delta_\theta\]
Supervised: Minimum Empirical Risk

- **Minimum Empirical Risk Training**

\[
\theta^* = \arg \min_{\theta} \sum_{x,y} \tilde{p}(x,y) L(\delta_\theta(x), y)
\]

- **MT decoder**

- **Empirical distribution**

- **x \rightarrow \delta_\theta \rightarrow \delta_\theta(x)\]**
Supervised: Minimum Empirical Risk

- **Minimum Empirical Risk Training**

$$\theta^* = \arg \min_{\theta} \sum_{x,y} \tilde{p}(x, y) L(\delta_\theta(x), y)$$

- **MT decoder**

- **Empirical distribution**

- **MT output**
Supervised: Minimum Empirical Risk

- **Minimum Empirical Risk Training**

\[
\theta^* = \arg\min_{\theta} \sum_{x,y} \tilde{p}(x,y)L(\delta_\theta(x), y)
\]

MT decoder

MT output

empirical distribution

loss

\[ x \rightarrow \delta_\theta \rightarrow \delta_\theta(x) \leftarrow y \]
Supervised: Minimum Empirical Risk

- **Minimum Empirical Risk Training**

\[
\theta^* = \arg \min_{\theta} \sum_{x,y} \tilde{p}(x, y) L(\delta_{\theta}(x), y)
\]

- MT decoder
- negated BLEU
- empirical distribution
- loss
- MT output
Supervised: Minimum Empirical Risk

- **Minimum Empirical Risk Training**

\[
\theta^* = \arg \min_{\theta} \sum_{x,y} \tilde{p}(x,y) L(\delta_\theta(x), y)
\]

- **Uniform Empirical Distribution**

\[
\theta^* = \arg \min_{\theta} \frac{1}{N} \sum_{i=1}^{N} L(\delta_\theta(x_i), \tilde{y}_i)
\]
Supervised: Minimum Empirical Risk

- **Minimum Empirical Risk Training**

\[ \theta^* = \arg \min_\theta \sum_{x,y} \tilde{p}(x,y) L(\delta_\theta(x), y) \]

MT decoder

negated BLEU

empirical distribution

- **Uniform Empirical Distribution**

\[ \theta^* = \arg \min_\theta \frac{1}{N} \sum_{i=1}^{N} L(\delta_\theta(x_i), \tilde{y}_i) \]

MT output

- MERT
- CRF
- Peceptron
Supervised: Minimum Empirical Risk

- **Minimum Empirical Risk Training**

\[ \theta^* = \arg \min_{\theta} \sum_{x,y} \tilde{p}(x,y) L(\delta_{\theta}(x), y) \]

- **Uniform Empirical Distribution**

\[ \theta^* = \arg \min_{\theta} \frac{1}{N} \sum_{i=1}^{N} L(\delta_{\theta}(x_i), \tilde{y}_i) \]

**What if the input x is missing?**

- MERT
- CRF
- Peceptron
Unsupervised: Minimum Imputed Risk

- Minimum **Empirical** Risk Training

\[
\theta^* = \arg \min_{\theta} \frac{1}{N} \sum_{i=1}^{N} L(\delta_\theta(x_i), \tilde{y}_i)
\]
Unsupervised: Minimum Imputed Risk

- **Minimum Empirical Risk Training**

\[
\theta^* = \arg\min_\theta \frac{1}{N} \sum_{i=1}^{N} L(\delta_\theta(x_i), \tilde{y_i})
\]

- **Minimum Imputed Risk Training**

\[
\theta^* = \arg\min_\theta \frac{1}{N} \sum_{i=1}^{N} \sum_x p_\phi(x \mid \tilde{y_i}) L(\delta_\theta(x), \tilde{y_i})
\]
Unsupervised: Minimum Imputed Risk

- **Minimum Empirical Risk Training**

\[
\theta^* = \arg \min_{\theta} \frac{1}{N} \sum_{i=1}^{N} L(\delta_{\theta}(x_i), \tilde{y}_i)
\]

- **Minimum Imputed Risk Training**

\[
\theta^* = \arg \min_{\theta} \frac{1}{N} \sum_{i=1}^{N} \sum_{x} p_{\phi}(x | \tilde{y}_i) L(\delta_{\theta}(x), \tilde{y}_i)
\]
Unsupervised: Minimum Imputed Risk

- **Minimum Empirical Risk Training**

\[ \theta^* = \arg\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} L(\delta_{\theta}(x_i), \tilde{y}_i) \]

- **Minimum Imputed Risk Training**

\[ \theta^* = \arg\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} \sum_{x} p(x | \tilde{y}_i) L(\delta_{\theta}(x), \tilde{y}_i) \]

\[ x \xrightarrow{\delta_{\theta}} \delta_{\theta}(x) \xleftarrow{\text{loss}} y \]
Unsupervised: Minimum Imputed Risk

- **Minimum Empirical Risk Training**

\[
\theta^* = \arg\min_\theta \frac{1}{N} \sum_{i=1}^{N} L(\delta_\theta(x_i), \tilde{y}_i)
\]

- **Minimum Imputed Risk Training**

\[
\theta^* = \arg\min_\theta \frac{1}{N} \sum_{i=1}^{N} \sum_{x} p_\phi(x \mid \tilde{y}_i) L(\delta_\theta(x), \tilde{y}_i)
\]
Unsupervised: Minimum Imputed Risk

- **Minimum Empirical Risk Training**

\[ \theta^* = \arg \min_{\theta} \frac{1}{N} \sum_{i=1}^{N} L(\delta_\theta(x_i), \tilde{y}_i) \]

- **Minimum Imputed Risk Training**

\[ \theta^* = \arg \min_{\theta} \frac{1}{N} \sum_{i=1}^{N} \sum_x p_\phi(x \mid \tilde{y}_i) L(\delta_\theta(x), \tilde{y}_i) \]

\( p_\phi \): reverse model
Unsupervised: Minimum Imputed Risk

- **Minimum Empirical Risk Training**

$$\theta^* = \arg \min_{\theta} \frac{1}{N} \sum_{i=1}^{N} L(\delta_\theta(x_i), \tilde{y}_i)$$

- **Minimum Imputed Risk Training**

$$\theta^* = \arg \min_{\theta} \frac{1}{N} \sum_{i=1}^{N} \sum_{x} p_\phi(x \mid \tilde{y}_i)L(\delta_\theta(x), \tilde{y}_i)$$

\(p_\phi\): reverse model
\(x\): imputed input
Unsupervised: Minimum Imputed Risk

- **Minimum Empirical Risk Training**

\[ \theta^* = \arg \min_{\theta} \frac{1}{N} \sum_{i=1}^{N} L(\delta_{\theta}(x_i), \tilde{y}_i) \]

- **Minimum Imputed Risk Training**

\[ \theta^* = \arg \min_{\theta} \frac{1}{N} \sum_{i=1}^{N} \sum_{x} p_{\phi}(x \mid \tilde{y}_i) L(\delta_{\theta}(x), \tilde{y}_i) \]

\( p_{\phi} \): reverse model
\( x \): imputed input
\( \delta_{\theta} \): forward system
Unsupervised: Minimum Imputed Risk

• Minimum Empirical Risk Training

$$\theta^* = \arg\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} L(\delta_{\theta}(x_i), \tilde{y}_i)$$

• Minimum Imputed Risk Training

$$\theta^* = \arg\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} \sum_{x} p_{\phi}(x | \tilde{y}_i) L(\delta_{\theta}(x), \tilde{y}_i)$$

$p_{\phi}$: reverse model
$x$: imputed input
$\delta_{\theta}$: forward system
Unsupervised: Minimum Imputed Risk

• **Minimum Empirical Risk Training**

\[
\theta^* = \arg \min_{\theta} \frac{1}{N} \sum_{i=1}^{N} L(\delta_{\theta}(x_i), \tilde{y}_i)
\]

• **Minimum Imputed Risk Training**

\[
\theta^* = \arg \min_{\theta} \frac{1}{N} \sum_{i=1}^{N} \sum_{x} p_\phi(x \mid \tilde{y}_i) L(\delta_{\theta}(x), \tilde{y}_i)
\]

\(p_\phi\): reverse model
\(x\): imputed input
\(\delta_{\theta}\): forward system

Round trip translation
Unsupervised: Minimum Imputed Risk

• Minimum **Empirical** Risk Training

\[ \theta^* = \arg \min_{\theta} \frac{1}{N} \sum_{i=1}^{N} L(\delta_\theta(x_i), \tilde{y}_i) \]

• Minimum **Imputed** Risk Training

\[ \theta^* = \arg \min_{\theta} \frac{1}{N} \sum_{i=1}^{N} \sum_x p_\phi(x | \tilde{y}_i) L(\delta_\theta(x), \tilde{y}_i) \]

\( p_\phi \): reverse model
\( \mathcal{X} \): imputed input
\( \delta_\theta \): forward system

Round trip translation
Speech recognition?
Chapter 4

We can also exploit both supervised and unsupervised data to perform semi-supervised training by using the reverse model. For each unsupervised example, we can use the reverse model to forward-translate each imputed target sentence $\bar{y}_i$ to the source language and back again. This process tries to ensure that probabilistic "round-trip" translation, from the source sentence to the target language sentence and back to the source sentence, will have a low expected loss. We will do this in our experiments.

Let us first review discriminative training in the reverse sense of optimizing the given performance metric. Our method is theoretically sound and can be explained as minimizing minimum imputed-risk, often better than the supervised case. Also, adding unsupervised data into the supervised training improves the performance.

For each unsupervised example $i$, we can use the reverse model $\phi$ to impute the missing source sentence $x_i$ as fol:

$$p_{\phi}(x_i) \rightarrow \delta_{\theta} \rightarrow \delta_{\theta}(x)$$

This process can be evaluated by looking more like true Chinese inputs. Note that $\theta$ is a kind of speech synthesizer that must produce a distribution over audio or at least over acoustic features or phone sequences. We will present several approximations in Section 4. We will do this in our experiments.
Our goal is to train a good forward system \( \delta_\theta \).
Our goal is to train a good forward system $\delta_\theta$

$p_\phi$ and $\delta_\theta$ are parameterized and trained separately
Our goal is to train a good forward system $\delta_\theta$.

$P_\phi$ and $\delta_\theta$ are parameterized and trained separately.

$P_\phi$ is fixed when training $\delta_\theta$.
Chapter 4

4.1 Minimum Empirical Risk (for Supervised Discrimination)

We perform experiments by using the open-source MT toolkit.

Specific to an MT task, our method works as follows. First guess forward, translate using a reverse English-to-Chinese model. Then train the discriminative model or a reverse translations of each imputed.

The crucial ingredient here is an interpolated version of a reverse prediction model that attempts to impute the missing data. One can manipulate the loss function to support other methods, such as Perceptron.

The trouble is that a typical reverse model often better the supervised case. Also, adding unsupervised data into the supervised training may have any form and need not be probabilistic.

Our minimum imputed-risk objective is a "reverse prediction model" that attempts to impute the missing result in the domain of some complex translation system. The method assumes that we are given some output data.

We can use audio or at least over acoustic features or phone sequences. For example, in a speech recognition task, we can use randomizing decoding.

Minimizing Imputed-Risk Training

One wishes to tune the parameters of the reverse translations strategy. Note that the randomization can be used in MERT.

Our method is also intuitive: it tries to ensure that probabilistic "roundtrip" translation often better the supervised case. Also, adding unsupervised data into the supervised training may have any form and need not be probabilistic.

Our method may be used for other tasks as well. For example, in a speech recognition task.
We perform experiments by using the open-source MT toolkit developed by Joshua Li et al. One wishes to tune the parameters of some complex translation system. In a translation task from English to Chinese, one usually does not make use of monolingual data as well as any reverse translations.

Our experiments show that unsupervised discriminative training performs similarly to and those discussed in Section 4.2.2 The Reverse Prediction Model. Now do ordinary supervised training as of the translations against the corresponding true translation.

The trouble is that a typical reverse model tries to ensure that probabilistic "round-trip" translation from the target language sentence to the source language and back again will have a low expected loss. In particular, it will generate a weighted lattice or hypergraph.

Our method is theoretically sound and can be explained as minimizing imputed risk. The crucial ingredient here is the reverse prediction model $\tilde{\phi}$ that attempts to impute the missing data to train a language model that will be useful for the reverse system. In particular, it will look more like true Chinese inputs.

We will present several approximations in Section 4.4. We will do this in our experiments. The loss of the translations against the corresponding true translation could be evaluated by an interpolated version of the reverse model.

One should not confuse this with the minimum risk training of $\phi$. For details see Chapter 4 from the target language sentence to the source language and back again will have low expected loss.
Chapter 4

Approximating $p_\phi(x \mid \tilde{y}_i)$

[Diagram]

- exponentially many $x$, stored in a hypergraph

SCFG

For each unsupervised example, we can exploit both supervised and unsupervised data to perform semi-supervised training by using...

We can also exploit both supervised and unsupervised data to perform semi-supervised training by using...

Our method is theoretically sound and can be explained as minimizing imputed risk.

Specific to an MT task, our method works as follows. First guess...

One wishes to tune the parameters using a reverse English to Chinese model. Then train the discriminative training method for discriminative training. Our method assumes that we are given some output data...

Our method may be used for other tasks as well. For example, in a speech recognition task, it will generate a weighted lattice or hyperlattice, as used in MERT.

Note that $\hat{\delta}_\theta$ is a “reverse prediction model” that attempts to impute the missing output data. Specific to our MT task, $\hat{\delta}_\theta$ is derived from the imputed training set. Specific to our MT task.

We will do this in our experiments.
Chapter 4

Our method may be used for other tasks as well. For example, in a speech recognition task.

Let us first review discriminative training in the conditional model $y \sim x$. One wishes to tune the parameters $(\theta, \phi)$ of the model to minimize the expected loss $\mathbb{E}[L(y, \hat{y})]$ over an unknown quantity $f_i$ replacing $L(y, \hat{y})$. Specific to our MT task, this is a differentiable function of the parameters $\theta, \phi$. If the loss function $L$ on page 4 is derived from the imputed training set $\tilde{y}_i$, specific to our MT task, it will have a low expected loss.

The trouble is that a typical reverse model is derived from the imputed training set $\tilde{y}_i$. Specific to our MT task, it will be probabilistic. For example, the parameters $\theta, \phi$ of the model that use deterministic decoding are not used.

However, this would not yield a differentiable objective function. Infinitesimal changes of the loss of the translations against the corresponding true translation are impractical.

One wishes to tune the parameters $(\theta, \phi)$ of the model to minimize the expected loss $\mathbb{E}[L(y, \hat{y})]$ over an unknown quantity $f_i$ replacing $L(y, \hat{y})$. Specific to our MT task, this is a differentiable function of the parameters $\theta, \phi$. If the loss function $L$ on page 4 is derived from the imputed training set $\tilde{y}_i$, specific to our MT task, it will have a low expected loss.

The trouble is that a typical reverse model is derived from the imputed training set $\tilde{y}_i$. Specific to our MT task, it will be probabilistic. For example, the parameters $\theta, \phi$ of the model that use deterministic decoding are not used.

One wishes to tune the parameters $(\theta, \phi)$ of the model to minimize the expected loss $\mathbb{E}[L(y, \hat{y})]$ over an unknown quantity $f_i$ replacing $L(y, \hat{y})$. Specific to our MT task, this is a differentiable function of the parameters $\theta, \phi$. If the loss function $L$ on page 4 is derived from the imputed training set $\tilde{y}_i$, specific to our MT task, it will have a low expected loss.

The trouble is that a typical reverse model is derived from the imputed training set $\tilde{y}_i$. Specific to our MT task, it will be probabilistic. For example, the parameters $\theta, \phi$ of the model that use deterministic decoding are not used.

One wishes to tune the parameters $(\theta, \phi)$ of the model to minimize the expected loss $\mathbb{E}[L(y, \hat{y})]$ over an unknown quantity $f_i$ replacing $L(y, \hat{y})$. Specific to our MT task, this is a differentiable function of the parameters $\theta, \phi$. If the loss function $L$ on page 4 is derived from the imputed training set $\tilde{y}_i$, specific to our MT task, it will have a low expected loss.

The trouble is that a typical reverse model is derived from the imputed training set $\tilde{y}_i$. Specific to our MT task, it will be probabilistic. For example, the parameters $\theta, \phi$ of the model that use deterministic decoding are not used.

One wishes to tune the parameters $(\theta, \phi)$ of the model to minimize the expected loss $\mathbb{E}[L(y, \hat{y})]$ over an unknown quantity $f_i$ replacing $L(y, \hat{y})$. Specific to our MT task, this is a differentiable function of the parameters $\theta, \phi$. If the loss function $L$ on page 4 is derived from the imputed training set $\tilde{y}_i$, specific to our MT task, it will have a low expected loss.

The trouble is that a typical reverse model is derived from the imputed training set $\tilde{y}_i$. Specific to our MT task, it will be probabilistic. For example, the parameters $\theta, \phi$ of the model that use deterministic decoding are not used.

One wishes to tune the parameters $(\theta, \phi)$ of the model to minimize the expected loss $\mathbb{E}[L(y, \hat{y})]$ over an unknown quantity $f_i$ replacing $L(y, \hat{y})$. Specific to our MT task, this is a differentiable function of the parameters $\theta, \phi$. If the loss function $L$ on page 4 is derived from the imputed training set $\tilde{y}_i$, specific to our MT task, it will have a low expected loss.

The trouble is that a typical reverse model is derived from the imputed training set $\tilde{y}_i$. Specific to our MT task, it will be probabilistic. For example, the parameters $\theta, \phi$ of the model that use deterministic decoding are not used.

One wishes to tune the parameters $(\theta, \phi)$ of the model to minimize the expected loss $\mathbb{E}[L(y, \hat{y})]$ over an unknown quantity $f_i$ replacing $L(y, \hat{y})$. Specific to our MT task, this is a differentiable function of the parameters $\theta, \phi$. If the loss function $L$ on page 4 is derived from the imputed training set $\tilde{y}_i$, specific to our MT task, it will have a low expected loss.

The trouble is that a typical reverse model is derived from the imputed training set $\tilde{y}_i$. Specific to our MT task, it will be probabilistic. For example, the parameters $\theta, \phi$ of the model that use deterministic decoding are not used.

One wishes to tune the parameters $(\theta, \phi)$ of the model to minimize the expected loss $\mathbb{E}[L(y, \hat{y})]$ over an unknown quantity $f_i$ replacing $L(y, \hat{y})$. Specific to our MT task, this is a differentiable function of the parameters $\theta, \phi$. If the loss function $L$ on page 4 is derived from the imputed training set $\tilde{y}_i$, specific to our MT task, it will have a low expected loss.

The trouble is that a typical reverse model is derived from the imputed training set $\tilde{y}_i$. Specific to our MT task, it will be probabilistic. For example, the parameters $\theta, \phi$ of the model that use deterministic decoding are not used.

One wishes to tune the parameters $(\theta, \phi)$ of the model to minimize the expected loss $\mathbb{E}[L(y, \hat{y})]$ over an unknown quantity $f_i$ replacing $L(y, \hat{y})$. Specific to our MT task, this is a differentiable function of the parameters $\theta, \phi$. If the loss function $L$ on page 4 is derived from the imputed training set $\tilde{y}_i$, specific to our MT task, it will have a low expected loss.

The trouble is that a typical reverse model is derived from the imputed training set $\tilde{y}_i$. Specific to our MT task, it will be probabilistic. For example, the parameters $\theta, \phi$ of the model that use deterministic decoding are not used.

One wishes to tune the parameters $(\theta, \phi)$ of the model to minimize the expected loss $\mathbb{E}[L(y, \hat{y})]$ over an unknown quantity $f_i$ replacing $L(y, \hat{y})$. Specific to our MT task, this is a differentiable function of the parameters $\theta, \phi$. If the loss function $L$ on page 4 is derived from the imputed training set $\tilde{y}_i$, specific to our MT task, it will have a low expected loss.

The trouble is that a typical reverse model is derived from the imputed training set $\tilde{y}_i$. Specific to our MT task, it will be probabilistic. For example, the parameters $\theta, \phi$ of the model that use deterministic decoding are not used.

One wishes to tune the parameters $(\theta, \phi)$ of the model to minimize the expected loss $\mathbb{E}[L(y, \hat{y})]$ over an unknown quantity $f_i$ replacing $L(y, \hat{y})$. Specific to our MT task, this is a differentiable function of the parameters $\theta, \phi$. If the loss function $L$ on page 4 is derived from the imputed training set $\tilde{y}_i$, specific to our MT task, it will have a low expected loss.

The trouble is that a typical reverse model is derived from the imputed training set $\tilde{y}_i$. Specific to our MT task, it will be probabilistic. For example, the parameters $\theta, \phi$ of the model that use deterministic decoding are not used.

One wishes to tune the parameters $(\theta, \phi)$ of the model to minimize the expected loss $\mathbb{E}[L(y, \hat{y})]$ over an unknown quantity $f_i$ replacing $L(y, \hat{y})$. Specific to our MT task, this is a differentiable function of the parameters $\theta, \phi$. If the loss function $L$ on page 4 is derived from the imputed training set $\tilde{y}_i$, specific to our MT task, it will have a low expected loss.

The trouble is that a typical reverse model is derived from the imputed training set $\tilde{y}_i$. Specific to our MT task, it will be probabilistic. For example, the parameters $\theta, \phi$ of the model that use deterministic decoding are not used.

One wishes to tune the parameters $(\theta, \phi)$ of the model to minimize the expected loss $\mathbb{E}[L(y, \hat{y})]$ over an unknown quantity $f_i$ replacing $L(y, \hat{y})$. Specific to our MT task, this is a differentiable function of the parameters $\theta, \phi$. If the loss function $L$ on page 4 is derived from the imputed training set $\tilde{y}_i$, specific to our MT task, it will have a low expected loss.

The trouble is that a typical reverse model is derived from the imputed training set $\tilde{y}_i$. Specific to our MT task, it will be probabilistic. For example, the parameters $\theta, \phi$ of the model that use deterministic decoding are not used.
Chapter 4

In this chapter, we describe an approach to machine translation (MT) called the reverse prediction model. The second step means that we must use a technique known as discriminative training, which involves training a machine translation system on a small amount of supervised data and then using this trained system to generate additional training data. This additional training data is then used to train a second machine translation system, which is used to generate additional training data, and so on. The trouble is that a typical reverse model may have any form and need not be probabilistic.

One wishes to tune the parameters of some complex translation system and add each reverse translation to forward, translate each imputed sentence to the target language, and then train the discriminator. Whereas one usually does not make use of monolingual data, one can exploit both supervised and unsupervised data to perform semi-supervised training by using a reverse prediction model that attempts to impute the missing data. The parameters of this model can then be used to tune the translation system.

Specific to an MT task, our method works as follows. First, guess the loss of the translations against the corresponding true translation. Then train the discriminator using a reverse English-to-Chinese model. The result is a "reverse prediction model" that attempts to impute the missing data. The resulting variant of this model is derived from the imputed training set. Specific to our MT task, the minimum imputed-risk objective is derived from the imputed training set. Our method may be used for other tasks as well. For example, in a speech recognition task, one may manipulate the loss function to support other probabilistic methods such as CRF.

Our method for discriminative training works as follows. Given an interpolated version of the model, we can also exploit both supervised and unsupervised data to perform semi-supervised training by using a reverse translation. The resulting variant of the model is derived from the imputed training set.

The diagram above illustrates the approximation of \( p_{\phi}(x | \tilde{y}_i) \). The x variables, stored in a hypergraph, are exponentially many and not closed under composition.

Approximating \( p_{\phi}(x | \tilde{y}_i) \)

\[
\begin{align*}
\delta_{\theta}(x) & \quad \tilde{y}_i \quad p_{\phi} \\
x & \quad \delta_{\theta}(x) \quad \tilde{y}_i \quad p_{\phi}
\end{align*}
\]
Approximating $p_\phi(x \mid \tilde{y}_i)$

exponentially many $x$, stored in a hypergraph

CFG is not closed under composition!

- **Approximations**
  - k-best
  - sampling
  - lattice
Chapter 4

For example, the parameters often better fit the supervised case. Also, adding unsupervised data into the supervised training may improve performance.

We perform experiments by using the open-source MT toolkit.

In particular, we use the reverse prediction model that attempts to impute the corresponding input data. This model in advance to forward, translate each imputed sample and add each reverse translation to the source language and back again will have a low expected loss.

Our minimum imputed-risk objective is derived from the imputed training set. Specific to our MT task, we can also exploit both supervised and unsupervised data to perform semi-supervised training by using a reverse prediction model that attempts to impute brute force decoding.

We suggest three approximations:
- k-best
- sampling
- lattice

CFG is not closed under composition!

• Approximations
  - k-best
  - sampling
  - lattice

variational approximation + lattice decoding (Dyer et al., 2008)
Our experiments show that unsupervised discriminative training performs similarly to 2 and \( \theta \) in the sense of optimizing the given performance metric.
The Forward System $\delta_\theta(x)$

- **Deterministic Decoding**
- **use one-best translation**

\[
\theta^* = \arg\min_\theta \frac{1}{N} \sum_{i=1}^{N} \sum_x p_\phi(x \mid \tilde{y}_i) L(\delta_\theta(x), \tilde{y}_i)
\]
The Forward System $\delta_\theta(x)$

- Deterministic Decoding
  - use **one-best** translation

$\theta^* = \arg\min_\theta \frac{1}{N} \sum_{i=1}^{N} \sum_x p_\phi(x \mid \tilde{y}_i) L(\delta_\theta(x), \tilde{y}_i)$

$\delta_\theta(x) = \arg\max_y p_\theta(y \mid x)$

the objective is not differentiable
The Forward System $\delta_\theta(x)$

- **Deterministic Decoding**
  - use **one-best** translation

$$\theta^* = \arg\min_\theta \frac{1}{N} \sum_{i=1}^{N} \sum_x p_\phi(x \mid \tilde{y}_i) L(\delta_\theta(x), \tilde{y}_i)$$

- **Randomized Decoding**
  - use a **distribution** of translations

$$\theta^* = \arg\min_\theta \frac{1}{N} \sum_{i=1}^{N} \sum_x p_\phi(x \mid \tilde{y}_i) \sum_y p_\theta(y \mid x) L(y, \tilde{y}_i)$$
The Forward System \( \delta_\theta(x) \)

- **Deterministic Decoding**
  - use **one-best** translation

\[
\theta^* = \arg\min_\theta \frac{1}{N} \sum_{i=1}^{N} \sum_x p_\phi(x | \tilde{y}_i) \sum_y \log \frac{p_\theta(y | x)}{\sum_{z} p_\theta(z | x)} L(\delta_\theta(x), \tilde{y}_i)
\]

- **Randomized Decoding**
  - use a **distribution** of translations

\[
\theta^* = \arg\min_\theta \frac{1}{N} \sum_{i=1}^{N} \sum_x p_\phi(x | \tilde{y}_i) \sum_y \sum_{z} p_\theta(z | x) L(y, \tilde{y}_i)
\]
The Forward System $\delta_\theta(x)$

- **Deterministic Decoding**
  - use **one-best** translation

  $$\theta^* = \arg\min_\theta \frac{1}{N} \sum_{i=1}^{N} \sum_x p_\phi(x \mid \tilde{y}_i) L(\delta_\theta(x), \tilde{y}_i)$$

- **Randomized Decoding**
  - use a **distribution** of translations

  $$\theta^* = \arg\min_\theta \frac{1}{N} \sum_{i=1}^{N} \sum_x p_\phi(x \mid \tilde{y}_i) \sum_y p_\theta(y \mid x) L(y, \tilde{y}_i)$$

The objective is not differentiable
In this chapter, we describe an approach to training translation systems. Specifically, in §4.1, we will introduce some notational conventions.

### Deterministic Decoding
- Use **one-best** translation

\[
\theta^* = \arg\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} \sum_{x} p_{\phi}(x | \tilde{y}_i) \mathcal{L}(\delta_{\theta}(x), \tilde{y}_i)
\]

### Randomized Decoding
- Use a **distribution** of translations

\[
\theta^* = \arg\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} \sum_{x} p_{\phi}(x | \tilde{y}_i) \sum_{y} p_{\theta}(y | x) \mathcal{L}(y, \tilde{y}_i)
\]
Experiments

- Supervised Training
  - require bitext

- Unsupervised Training
  - require monolingual English

- Semi-supervised Training
  - interpolation of supervised and unsupervised
Semi-supervised Training
## Semi-supervised Training

<table>
<thead>
<tr>
<th>Training scenario</th>
<th>Test BLEU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sup, (200, 200*16)</td>
<td>47.6</td>
</tr>
</tbody>
</table>
Semi-supervised Training

<table>
<thead>
<tr>
<th>Training scenario</th>
<th>Test BLEU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sup, (200, 200*16)</td>
<td>47.6</td>
</tr>
</tbody>
</table>

### Table

<table>
<thead>
<tr>
<th>Data size</th>
<th>Test BLEU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sup</td>
<td>Unsup</td>
</tr>
<tr>
<td>5 sentences, 7 references</td>
<td>5 sentences, 7 references</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Unsup</th>
<th>Sup</th>
</tr>
</thead>
<tbody>
<tr>
<td>5 sentences, 7 references</td>
<td>5 sentences, 7 references</td>
</tr>
</tbody>
</table>

![Diagram showing the training scenarios and test BLEU scores.](Diagram)

## 4.4.5 Supervised and Unsupervised Training

Though the semi-supervised training scenario is the most likely scenario in practice, we are interested in knowing how our unsupervised discriminative method performs when compared with supervised discriminative training under different training data sizes. Table (.) shows relevant results. Surprisingly, our unsupervised method actually performs better than the supervised training in most cases. Also, using more unsupervised data is better in general.

A possible explanation for the high performance of the unsupervised method is that the training set is considerably larger and more diverse in this case. Whereas the supervised method learns to translate only a single observed \( x_i \) sentence into something resembling the \( \tilde{y}_i \) set, the unsupervised method learns to translate each of the imputed \( x_i \) sentences back into something resembling the \( \tilde{y}_i \) set (as explained above). These imputed \( x_i \) sentences can use different vocabulary and sentence structure derived from the different references.
Semi-supervised Training

<table>
<thead>
<tr>
<th>Training scenario</th>
<th>Test BLEU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sup, (200, 200*16)</td>
<td>47.6</td>
</tr>
</tbody>
</table>

40K sent. pairs

Table (.-: BLEU scores for semi-supervised training.

The supervised system 5"Sup"6 is trained on a bilingual data set having 9 Chinese sentences and 7" English references. "8Unsup" means that we add unsupervised data 9i.e.9 monolingual English sentences 9 for training. For each English sentence 9 we impute a one-best Chinese translation using the reverse translation system.

Table (.-: BLEU scores for supervised and unsupervised training.

For supervised training 5"Sup"6 a data size of 9 means that we use a bilingual data set having 9 Chinese sentences and 7" English references. For unsupervised training 5"Unsup"6 a data size of 9 means that we use an English monolingual data set having 7" sentences 9 and for each English sentence we will impute a one-best Chinese translation by using the reverse translation system.

4.4.5 Supervised and Unsupervised Training

Though the semi-supervised training scenario is the most likely scenario in practice, we are interested in knowing how our unsupervised discriminative method performs when compared with supervised discriminative training, under different training data sizes. Table (-.) shows relevant results. Surprisingly, our unsupervised method actually performs better than the supervised training in most cases. Also, using more unsupervised data is better in general.

A possible explanation for the high performance of the unsupervised method is that the training set is considerably larger and more diverse in this case. Whereas the supervised method learns to translate only a single observed sentence into something resembling the set, the unsupervised method learns to translate each of the imputed sentences back into something resembling the set (as explained above). These imputed sentences can use different vocabulary and sentence structure derived from the different references.

-
4.4.5 Supervised and Unsupervised Training

Though the semi-supervised training scenario is the most likely scenario in practice, we are interested in knowing how our unsupervised discriminative method performs when compared with supervised discriminative training under different training data sizes. Table shows relevant results. Surprisingly, our unsupervised method actually performs better than the supervised training in most cases. Also, using more unsupervised data is better in general.

A possible explanation for the high performance of the unsupervised method is that the training set is considerably larger and more diverse in this case. Whereas the supervised method learns to translate only a single observed \( x_i \) sentence into something resembling the \( \tilde{y}_i \) set, the unsupervised method learns to translate each of the imputed \( x_i \) sentences back into something resembling the \( \tilde{y}_i \) set (as explained above). These imputed \( x_i \) sentences can use different vocabulary and sentence structure derived from the different references.

Table: BLEU scores for supervised and unsupervised training.

<table>
<thead>
<tr>
<th>Training scenario</th>
<th>Test BLEU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sup, (200, 200*16)</td>
<td>47.6</td>
</tr>
</tbody>
</table>

Table: BLEU scores for semi-supervised training.

<table>
<thead>
<tr>
<th>Training scenario</th>
<th>Test BLEU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sup, (200, 200*16)</td>
<td>47.6</td>
</tr>
</tbody>
</table>

Table (blank): BLEU scores for semi-supervised training.

<table>
<thead>
<tr>
<th>Training scenario</th>
<th>Test BLEU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sup, (200, 200*16)</td>
<td>47.6</td>
</tr>
</tbody>
</table>

Table (blank): BLEU scores for semi-supervised training.

<table>
<thead>
<tr>
<th>Training scenario</th>
<th>Test BLEU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sup, (200, 200*16)</td>
<td>47.6</td>
</tr>
</tbody>
</table>
Semi-supervised Training

<table>
<thead>
<tr>
<th>Training scenario</th>
<th>Test BLEU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sup, (200, 200*16)</td>
<td>47.6</td>
</tr>
</tbody>
</table>

Table (.-: BLEU scores for semi-supervised training.

For supervised training ("Sup") a data size of " means that we use a bilingual data set having " Chinese sentences and " English references. For unsupervised training ("Unsup") a data size of " means that we use an English monolingual data set having " sentences and for each English sentence we will impute a one-best Chinese translation by using the reverse translation system.

4.4.5 Supervised and Unsupervised Training

Though the semi-supervised training scenario is the most likely scenario in practice, we are interested in knowing how our unsupervised discriminative method performs when compared with supervised discriminative training, under different training data sizes. Table (.-: shows relevant results. Surprisingly, our unsupervised method actually performs better than the supervised training in most cases. Also, using more unsupervised data is better in general.

A possible explanation for the high performance of the unsupervised method is that the training set is considerably larger and more diverse in this case. Whereas the supervised method learns to translate only a single observed \( x_i \) sentence into something resembling the \( \tilde{y}_i \) set, the unsupervised method learns to translate each of the imputed \( x_i \) sentences back into something resembling the \( \tilde{y}_i \) set (as explained above). These imputed \( x_i \) sentences can use different vocabulary and sentence structure derived from the different references.

40K sent. pairs

551 features
Semi-supervised Training

<table>
<thead>
<tr>
<th>Training scenario</th>
<th>Test BLEU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sup, (200, 200*16)</td>
<td>47.6</td>
</tr>
<tr>
<td>+Unsup, 100*16 Eng sentences</td>
<td>49.0</td>
</tr>
<tr>
<td>+Unsup, 200*16 Eng sentences</td>
<td>48.9</td>
</tr>
</tbody>
</table>

40K sent. pairs

Adding unsupervised data helps!

551 features
Supervised vs. Unsupervised

Unsupervised training performs as well as (and often better than) the supervised one!
Unsupervised training performs as well as (and often better than) the supervised one!

Unsupervised uses 16 times of data as supervised. For example,
Unsupervised training performs as well as (and often better than) the supervised one!

Unsupervised uses 16 times of data as supervised. For example,

<table>
<thead>
<tr>
<th></th>
<th>Chinese</th>
<th>English</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sup</td>
<td>100</td>
<td>16*100</td>
</tr>
</tbody>
</table>
Supervised vs. Unsupervised

Unsupervised training performs as well as (and often better than) the supervised one!

Unsupervised uses 16 times of data as supervised. For example,

<table>
<thead>
<tr>
<th></th>
<th>Chinese</th>
<th>English</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sup</td>
<td>100</td>
<td>16*100</td>
</tr>
<tr>
<td>Unsup</td>
<td>16*100</td>
<td>16<em>16</em>100</td>
</tr>
</tbody>
</table>
Supervised vs. Unsupervised

Unsupervised training performs as well as (and often better than) the supervised one!

Unsupervised uses 16 times of data as supervised. For example,

<table>
<thead>
<tr>
<th></th>
<th>Chinese</th>
<th>English</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sup</td>
<td>100</td>
<td>16*100</td>
</tr>
<tr>
<td>Unsup</td>
<td>16*100</td>
<td>16<em>16</em>100</td>
</tr>
</tbody>
</table>

But, fair comparison!
Unsupervised training performs as well as (and often better than) the supervised one! Unsupervised uses 16 times of data as supervised. For example,

<table>
<thead>
<tr>
<th></th>
<th>Chinese</th>
<th>English</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sup</td>
<td>100</td>
<td>16*100</td>
</tr>
<tr>
<td>Unsup</td>
<td>16*100</td>
<td>16<em>16</em>100</td>
</tr>
</tbody>
</table>

But, fair comparison!

- More experiments
Unsupervised training performs as well as (and often better than) the supervised one!

Unsupervised uses 16 times of data as supervised. For example,

<table>
<thead>
<tr>
<th></th>
<th>Chinese</th>
<th>English</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sup</td>
<td>100</td>
<td>16*100</td>
</tr>
<tr>
<td>Unsup</td>
<td>16*100</td>
<td>16<em>16</em>100</td>
</tr>
</tbody>
</table>

But, fair comparison!

- More experiments
  - different k-best size
Supervised vs. Unsupervised

Unsupervised training performs as well as (and often better than) the supervised one!

Unsupervised uses **16** times of data as supervised. For example,

<table>
<thead>
<tr>
<th></th>
<th>Chinese</th>
<th>English</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Sup</strong></td>
<td>100</td>
<td>16*100</td>
</tr>
<tr>
<td><strong>Unsup</strong></td>
<td>16*100</td>
<td>16<em>16</em>100</td>
</tr>
</tbody>
</table>

But, fair comparison!

- More experiments
  - different k-best size
  - different reverse model
Outline

• Hypergraph as Hypothesis Space
• Unsupervised Discriminative Training
  ▸ minimum imputed risk
  ▸ contrastive language model estimation
• Variational Decoding
• First- and Second-order Expectation Semirings
Language Modeling
Language Modeling

- **Language Model** $p_{\theta}(y)$
  - assign a probability to an English sentence $y$
  - typically use an $n$-gram model

5.1 Unsupervised Training of Global Log-Linear Language Models

We have a set of training examples $\tilde{y}_i$ where each $\tilde{y}_i$ is a sequence of English words. We aim to train a language model $p_{\theta}(y)$ parameterized by $\theta$ over the examples. The model will be used to assign a probability to any English sentence. We can obtain such a model by maximizing the likelihood of the training examples as follows:

$$\theta^* = \arg \max_{\theta} \prod_{i} p_{\theta}(\tilde{y}_i)$$

Now the question is: what is the form of $p_{\theta}(\tilde{y}_i)$?

5.1.1 Whole-sentence Maximum Entropy Language Model

We can specify $p_{\theta}$ as a globally normalized log-linear model as follows:

$$p_{\theta}(y) = e^{f(y) \cdot \theta} / Z(\theta)$$

where $f(y)$ is a feature vector depending on $y$ and $\theta$ is the corresponding weight vector. This is called as a whole-sentence maximum entropy language model.

Training with the above log-linear model requires computing the normalization constant $Z(\theta)$ alternatively called the partition function, which is computationally challenging as it requires to sum over all the possible sequences $y \in \Sigma^*$. In our case, this corresponds to computing the sum over all possible English sentences with any length.

To address the computational difficulty issue, Rosenfeld et al. approximately compute $Z(\theta)$ by using a set of sentences that are sampled from $\Sigma^*$.

5.1.2 Contrastive Estimation

Smith and Eisner propose contrastive estimation (CE) which uses a small neighborhood instead of the full space $\Sigma^*$ to approximate the computation of normalization constant. Under CE, the likelihood of an observed sequence $\tilde{y}$ is:

$$p_{\theta}(\tilde{y}) = e^{f(\tilde{y}) \cdot \theta} / Z(\tilde{y})$$

One should not confuse this with a regular maximum entropy language model where the normalization is done for each $n$-gram history, i.e., locally normalized.
Language Modeling

- **Language Model** $p_\theta(y)$
- assign a probability to an English sentence $y$
- typically use an $n$-gram model

$$p(y) = \prod_{w \in W_n} p(r(w) \mid h(w))^{c_w(y)}$$
Language Modeling

- **Language Model** $p_\theta(y)$
  - assign a probability to an English sentence $y$
  - typically use an n-gram model

$$p(y) = \prod_{w \in W_n} p(r(w) \mid h(w))^{c_w(y)}$$

a set of n-grams occurred in $y$
Language Modeling

- Language Model \( p_\theta(y) \)
- assign a probability to an English sentence \( y \)
- typically use an n-gram model

\[
p(y) = \prod_{w \in W_n} p(r(w) \mid h(w))^{c_w(y)}
\]

Locally normalized

a set of n-grams occurred in \( y \)
Language Modeling

- **Language Model** $p_\theta(y)$
- assign a probability to an English sentence $y$
- typically use an $n$-gram model

\[
p(y) = \prod_{w \in W_n} p(r(w) \mid h(w))^c_w(y)
\]

Locally normalized

- **Global Log-linear Model**
  (whole-sentence maximum-entropy LM)

\[
p_\theta(y) = \frac{e^{f(y) \cdot \theta}}{Z(*)}
\]

(Rosenfeld et al., 2001)
Language Modeling

- **Language Model** \( p_\theta(y) \)
- assign a probability to an English sentence \( y \)
- typically use an n-gram model

\[
p(y) = \prod_{w \in W_n} p(r(w) \mid h(w))^c_w(y) \]

Locally normalized

- **Global Log-linear Model**
  (whole-sentence maximum-entropy LM)

\[
p_\theta(y) = \frac{e^{f(y) \cdot \theta}}{Z(\ast)}
\]

Globally normalized

(Rosenfeld et al., 2001)
Language Modeling

- Language Model $p_\theta(y)$
- assign a probability to an English sentence $y$
- typically use an n-gram model

$$p(y) = \prod_{w \in W_n} p(r(w) \mid h(w))^{c_w(y)}$$

Locally normalized

- Global Log-linear Model
  (whole-sentence maximum-entropy LM)

$$(\text{Rosenfeld et al., 2001})$$

$$p_\theta(y) = \frac{e^{f(y) \cdot \theta}}{Z(\ast)}$$

Globally normalized

$$Z(\ast) \overset{\text{def}}{=} \sum_{y' \in \Sigma^*} e^{f(y') \cdot \theta}$$
Language Modeling

- **Language Model** \( p_\theta(y) \)
  - assign a probability to an English sentence \( y \)
  - typically use an n-gram model

\[
p(y) = \prod_{w \in W_n} p(r(w) \mid h(w))^f_w(y)
\]

Locally normalized

- **Global Log-linear Model** (whole-sentence maximum-entropy LM)
  (Rosenfeld et al., 2001)

\[
p_\theta(y) = \frac{e^{f(y)} \cdot \theta}{Z(\ast)}
\]

Globally normalized

\[
Z(\ast) \overset{\text{def}}{=} \sum_{y' \in \Sigma^*} e^{f(y')} \cdot \theta
\]

All English sentences with any length!
Language Modeling

- **Language Model** $p_\theta(y)$
  - assign a probability to an English sentence $y$
  - typically use an n-gram model

\[
p(y) = \prod_{w \in W_n} p(r(w) | h(w))^{c_w(y)}
\]

Locally normalized

- **Global Log-linear Model**
  - (whole-sentence maximum-entropy LM)
  - (Rosenfeld et al., 2001)

\[
p_\theta(y) = \frac{e^{f(y) \cdot \theta}}{Z(\ast)}
\]

Globally normalized

\[
Z(\ast) \overset{\text{def}}{=} \sum_{y' \in \Sigma^*} e^{f(y') \cdot \theta}
\]

All English sentences with any length!

**Sampling**
Language Modeling

- **Language Model** $p_\theta(y)$
  - assign a probability to an English sentence $y$
  - typically use an n-gram model

$$p(y) = \prod_{w \in W_n} p(r(w) \mid h(w))^{|w(y)|}$$

Locally normalized

- **Global Log-linear Model**
  - (whole-sentence maximum-entropy LM)

$$p_\theta(y) = \frac{e^{f(y) \cdot \theta}}{Z(*)}$$

Globally normalized

All English sentences with any length!

**Sampling** slow 😞
Contrastive Estimation

- Global Log-linear Model
  (whole-sentence maximum-entropy LM) (Rosenfeld et al., 2001)

\[ p_\theta(y) = \frac{e^{f(y) \cdot \theta}}{Z(*)} \]

\[ Z(*) \overset{\text{def}}{=} \sum_{y' \in \Sigma^*} e^{f(y') \cdot \theta} \]
Contrastive Estimation

- **Global Log-linear Model**
  
  (whole-sentence maximum-entropy LM) (Rosenfeld et al., 2001)

  \[
  p_{\theta}(y) = \frac{e^{f(y) \cdot \theta}}{Z(\star)}
  \]

  \[
  Z(\star) \overset{\text{def}}{=} \sum_{y' \in \Sigma^\star} e^{f(y') \cdot \theta}
  \]

- **Contrastive Estimation (CE)** (Smith and Eisner, 2005)

  \[
  p_{\theta}(\tilde{y}) = \frac{e^{f(\tilde{y}) \cdot \theta}}{Z(\tilde{y})} = \frac{e^{f(\tilde{y}) \cdot \theta}}{\sum_{y' \in \mathcal{N}(\tilde{y})} e^{f(y') \cdot \theta}}
  \]

- The question is: what is the form of it requires to sum over all the possible sequence if it requires to sum over all the possible sequence
**Contrastive Estimation**

- **Global Log-linear Model** *(whole-sentence maximum-entropy LM)*  
  
  \[
  p_\theta(y) = \frac{e^{f(y)\cdot\theta}}{Z(*)} 
  \]

- **Contrastive Estimation (CE)** *(Smith and Eisner, 2005)*  
  
  \[
  p_\theta(\tilde{y}) = \frac{e^{f(\tilde{y})\cdot\theta}}{Z(\tilde{y})} = \frac{e^{f(\tilde{y})\cdot\theta}}{\sum_{y' \in \mathcal{N}(\tilde{y})} e^{f(y')\cdot\theta}}
  \]

\[
Z(*) \overset{\text{def}}{=} \sum_{y' \in \Sigma^*} e^{f(y')\cdot\theta}
\]

(Rosenfeld et al., 2001)
Contrastive Estimation

- **Global Log-linear Model** *(whole-sentence maximum-entropy LM)*

  \[
  p_\theta(y) = \frac{e^{f(y)} \cdot \theta}{Z(\ast)} 
  \]

  \[
  Z(\ast) \overset{\text{def}}{=} \sum_{y' \in \Sigma^*} e^{f(y')} \cdot \theta
  \]

- **Contrastive Estimation (CE)** *(Smith and Eisner, 2005)*

  \[
  p_\theta(\tilde{y}) = \frac{e^{f(\tilde{y})} \cdot \theta}{Z(\tilde{y})} = \frac{e^{f(\tilde{y})} \cdot \theta}{\sum_{y' \in \mathcal{N}(\tilde{y})} e^{f(y')} \cdot \theta}
  \]

\[
\tilde{y}
\]
### Contrastive Estimation

- **Global Log-linear Model** *(whole-sentence maximum-entropy LM)*

\[
p_{\theta}(y) = \frac{e^{f(y) \cdot \theta}}{Z(*)}
\]

where \( Z(*) \) is a normalization constant. Under CE, the likelihood of an observed sequence will be used to assign a probability to any English sentence. We can obtain such a model as a globally normalized log-linear model as follows:

\[
Z(*) \overset{\text{def}}{=} \sum_{y' \in \Sigma^*} e^{f(y') \cdot \theta}
\]

- **Contrastive Estimation (CE)** *(Smith and Eisner, 2005)*

\[
p_{\theta}(\tilde{y}) = \frac{e^{f(\tilde{y}) \cdot \theta}}{Z(\tilde{y})} = \frac{e^{f(\tilde{y}) \cdot \theta}}{\sum_{y' \in \mathcal{N}(\tilde{y})} e^{f(y') \cdot \theta}}
\]

\( \tilde{y} \) is a sequence of English words, \( \mathcal{N}(\tilde{y}) \) is the corresponding weight vector, and \( \theta \) is a parameterized by \( f(\cdot) \).

- **Contrastive Estimation (CE) (Rosenfeld et al., 2001)**

\[
\tilde{y} \rightarrow \text{Neighborhood Function} \rightarrow \mathcal{N}(\tilde{y})
\]
Contrastive Estimation

- **Global Log-linear Model**
  - *(whole-sentence maximum-entropy LM)*
  
  \[
  p_\theta(y) = \frac{e^{f(y)} \cdot \theta}{Z(*)}
  \]

- **Contrastive Estimation (CE)**
  - *(Smith and Eisner, 2005)*
  \[
  p_\theta(\tilde{y}) = \frac{e^{f(\tilde{y})} \cdot \theta}{Z(\tilde{y})} = \frac{e^{f(\tilde{y})} \cdot \theta}{\sum_{y' \in \mathcal{N}(\tilde{y})} e^{f(y')} \cdot \theta}
  \]

\(\tilde{y}\) → Neighborhood Function → \(\mathcal{N}(\tilde{y})\)

neighborhood or contrastive set
**Contrastive Estimation**

- **Global Log-linear Model** *(whole-sentence maximum-entropy LM)*

\[
p_\theta(y) = \frac{e^{f(y) \cdot \theta}}{Z(*)}\]

\[
Z(*) \overset{\text{def}}{=} \sum_{y' \in \Sigma^*} e^{f(y') \cdot \theta}
\]

- **Contrastive Estimation (CE)** *(Smith and Eisner, 2005)*

\[
p_\theta(\tilde{y}) = \frac{e^{f(\tilde{y}) \cdot \theta}}{Z(\tilde{y})} = \frac{e^{f(\tilde{y}) \cdot \theta}}{\sum_{y' \in \mathcal{N}(\tilde{y})} e^{f(y') \cdot \theta}}
\]

\[
\tilde{y} \rightarrow \text{Neighborhood Function} \rightarrow \mathcal{N}(\tilde{y})
\]

*a set of alternate Eng. sentences of \( \tilde{y} \) neighborhood or contrastive set*
**Contrastive Estimation**

• **Global Log-linear Model**
  (whole-sentence maximum-entropy LM)  
  \[
  p_\theta(y) = \frac{e^{f(y) \cdot \theta}}{Z(\ast)}
  \]
  \[
  Z(\ast) \overset{\text{def}}{=} \sum_{y' \in \Sigma^*} e^{f(y') \cdot \theta}
  \]

• **Contrastive Estimation (CE)**  
  \[
  p_\theta(\tilde{y}) = \frac{e^{f(\tilde{y}) \cdot \theta}}{Z(\tilde{y})} = \sum_{y' \in N(\tilde{y})} e^{f(y') \cdot \theta}
  \]
  \[
  \text{loss}
  \]

  \[
  \tilde{y} \rightarrow \text{Neighborhood Function} \rightarrow N(\tilde{y})
  \]

  a set of alternate Eng. sentences of \(\tilde{y}\)

  neighborhood or contrastive set
**Contrastive Estimation**

- **Global Log-linear Model**
  (whole-sentence maximum-entropy LM)  
  \[
  p_\theta(y) = \frac{e^{f(y) \cdot \theta}}{Z(\star)}
  \]
  \[
  Z(\star) \overset{\text{def}}{=} \sum_{y' \in \Sigma^*} e^{f(y') \cdot \theta}
  \]

- **Contrastive Estimation (CE)**  
  \[
  p_\theta(\tilde{y}) = \frac{e^{f(\tilde{y}) \cdot \theta}}{Z(\tilde{y})} = \frac{e^{f(\tilde{y}) \cdot \theta}}{\sum_{y' \in \mathcal{N}(\tilde{y})} e^{f(y') \cdot \theta}}
  \]

---

**Smith and Eisner, 2005**

**Rosenfeld et al., 2001**
Contrastive Estimation

- **Global Log-linear Model**
  (whole-sentence maximum-entropy LM)

\[
p_\theta(y) = \frac{e^{f(y)} \cdot \theta}{Z(*)}
\]

- **Contrastive Estimation (CE)**

\[
p_\theta(\tilde{y}) = \frac{e^{f(\tilde{y})} \cdot \theta}{Z(\tilde{y})} = \frac{e^{f(\tilde{y})} \cdot \theta}{\sum_{y' \in \mathcal{N}(\tilde{y})} e^{f(y')} \cdot \theta}
\]

(Rosenfeld et al., 2001)

(Smith and Eisner, 2005)

 improves both speed and accuracy
Contrastive Estimation

- **Global Log-linear Model**
  (whole-sentence maximum-entropy LM)

\[
p_\theta(y) = \frac{e^{f(y) \cdot \theta}}{Z(*)}
\]

\[
Z(*) \overset{\text{def}}{=} \sum y' \in \Sigma* e^{f(y') \cdot \theta}
\]

- **Contrastive Estimation (CE)**

\[
p_\theta(\tilde{y}) = \frac{e^{f(\tilde{y}) \cdot \theta}}{Z(\tilde{y})} = \frac{e^{f(\tilde{y}) \cdot \theta}}{\sum y' \in \mathcal{N}(\tilde{y}) e^{f(y') \cdot \theta}}
\]

Improve both speed and accuracy

Not proposed for language modeling

\(\tilde{y}\) → Neighborhood Function → \(\mathcal{N}(\tilde{y})\)

Loss

\(\tilde{y}\)
a set of alternate Eng. sentences of \(\tilde{y}\)

neighboring or contrastive set

(Rosenfeld et al., 2001)

(Smith and Eisner, 2005)
Contrastive Estimation

- **Global Log-linear Model**
  (whole-sentence maximum-entropy LM)
  \[ p_{\theta}(y) = \frac{e^{f(y)\cdot \theta}}{Z(\star)} \]
  \[ Z(\star) \overset{\text{def}}{=} \sum_{y' \in \Sigma} e^{f(y')\cdot \theta} \]

- **Contrastive Estimation (CE)**
  \[ p_{\theta}(\tilde{y}) = \frac{e^{f(\tilde{y})\cdot \theta}}{Z(\tilde{y})} = \frac{e^{f(\tilde{y})\cdot \theta}}{\sum_{y' \in \mathcal{N}(\tilde{y})} e^{f(y')\cdot \theta}} \]

\[ \mathcal{N}(\tilde{y}) \]

- to approximate the computation of normalization
- not proposed for language modeling
- train to recover the original English as much as possible
- improve both speed and accuracy

---

4 propose contrastive estimation (CE) which uses a small neighborhood or contrastive set

\[ \text{loss} \]

\[ \tilde{y} \rightarrow \text{Neighborhood Function} \rightarrow \mathcal{N}(\tilde{y}) \]

a set of alternate English sentences of \( \tilde{y} \)

(Rosenfeld et al., 2001)

(Smith and Eisner, 2005)
Contrastive Language Model Estimation
Contrastive Language Model Estimation

- **Step-1**: extract a confusion grammar (CG)
- an English-to-English SCFG
Contrastive Language Model Estimation

- **Step-1**: extract a confusion grammar \((CG)\)
- an English-to-English SCFG

neighborhood function
Contrastive Language Model Estimation

- **Step-1**: extract a confusion grammar (CG)
  - an English-to-English SCFG

- **Step-2**: for each English sentence, generate a contrastive set (or neighborhood) using the CG
Contrastive Language Model Estimation

- **Step-1**: extract a **confusion grammar** (CG)
- an English-to-English SCFG

- **Step-2**: for each English sentence, generate a **contrastive set** (or **neighborhood**) using the CG

- **Step-3**: discriminative training
Contrastive Language Model Estimation

- **Step-1**: extract a **confusion grammar** (CG)
  - an English-to-English SCFG

- **Step-2**: for each English sentence, generate a **contrastive set** (or **neighborhood**) using the CG

- **Step-3**: **discriminative training**
Contrastive Language Model Estimation

- **Step-1**: extract a **confusion grammar** (CG)
- an English-to-English SCFG

\[
X \rightarrow \langle \text{lead to, result in} \rangle
\]

- **Step-2**: for each English sentence, generate a **contrastive set** (or **neighborhood**) using the CG

- **Step-3**: discriminative training
Contrastive Language Model Estimation

- **Step-1**: extract a **confusion grammar** (CG)
  - an English-to-English SCFG
  
  \[ X \rightarrow \langle \text{lead to, result in} \rangle \]

- **Step-2**: for each English sentence, generate a **contrastive set** (or **neighborhood**) using the CG

- **Step-3**: discriminative training
Contrastive Language Model Estimation

- **Step-1**: extract a confusion grammar (CG)
- an English-to-English SCFG

\[
X \rightarrow \langle \text{lead to, result in} \rangle \\
X \rightarrow \langle X_0 \text{ at beijing, beijing 's } X_0 \rangle
\]

- **Step-2**: for each English sentence, generate a contrastive set (or neighborhood) using the CG

- **Step-3**: discriminative training
Contrastive Language Model Estimation

- **Step-1:** extract a confusion grammar (CG)
  - an English-to-English SCFG
    
    \[
    X \rightarrow \left\langle \text{lead to, result in} \right\rangle \\
    X \rightarrow \left\langle X_0 \text{ at beijing, beijing 's } X_0 \right\rangle \\
    X \rightarrow \left\langle X_0 \text{ of } X_1, X_0 \text{ of the } X_1 \right\rangle
    \]

- **Step-2:** for each English sentence, generate a contrastive set (or neighborhood) using the CG

- **Step-3:** discriminative training
Contrastive Language Model Estimation

- **Step-1**: extract a confusion grammar (CG)
  - an English-to-English SCFG

  \[
  X \rightarrow \langle \text{lead to, result in} \rangle \\
  X \rightarrow \langle X_0 \text{ at beijing, beijing 's } X_0 \rangle \\
  X \rightarrow \langle X_0 \text{ of } X_1, X_0 \text{ of the } X_1 \rangle 
  \]

- **Step-2**: for each English sentence, generate a contrastive set (or neighborhood) using the CG

- **Step-3**: discriminative training
Contrastive Language Model Estimation

- **Step-1:** extract a **confusion grammar** (CG)
  - an English-to-English SCFG
    
    \[
    X \rightarrow \langle \text{lead to}, \text{result in} \rangle
    \]
    
    \[
    X \rightarrow \langle X_0 \text{ at beijing, beijing 's } X_0 \rangle
    \]
    
    \[
    X \rightarrow \langle X_0 \text{ of } X_1, X_0 \text{ of the } X_1 \rangle
    \]
    
    \[
    X \rightarrow \langle X_0 \text{ 's } X_1, X_1 \text{ of } X_0 \rangle
    \]

- **Step-2:** for each English sentence, generate a **contrastive set** (or **neighborhood**) using the CG

- **Step-3:** discriminative training
**Contrastive Language Model Estimation**

- **Step-1:** extract a confusion grammar (CG)
  - an English-to-English SCFG
    
    \[ X \rightarrow \langle \text{lead to, result in} \rangle \]
    
    \[ X \rightarrow \langle X_0 \text{ at beijing, beijing 's } X_0 \rangle \]
    
    \[ X \rightarrow \langle X_0 \text{ of } X_1, X_0 \text{ of the } X_1 \rangle \]
    
    \[ X \rightarrow \langle X_0 \text{ 's } X_1, X_1 \text{ of } X_0 \rangle \]

- **Step-2:** for each English sentence, generate a contrastive set (or neighborhood) using the CG

- **Step-3:** discriminative training
Step-1: Extracting a Confusion Grammar (CG)
Step 1: Extracting a Confusion Grammar (CG)

- Deriving a CG from a bilingual grammar
  - use Chinese side as pivots
Step-1: Extracting a Confusion Grammar (CG)

- Deriving a CG from a bilingual grammar
  - use Chinese side as pivots

Bilingual Rule | Confusion Rule
Step 1: Extracting a Confusion Grammar (CG)

- Deriving a CG from a bilingual grammar
- use Chinese side as pivots

<table>
<thead>
<tr>
<th>Bilingual Rule</th>
<th>Confusion Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X \rightarrow \langle \text{mao, a cat} \rangle$</td>
<td></td>
</tr>
<tr>
<td>$X \rightarrow \langle \text{mao, the cat} \rangle$</td>
<td></td>
</tr>
</tbody>
</table>
Step 1: Extracting a Confusion Grammar (CG)

- Deriving a CG from a bilingual grammar
- Use Chinese side as pivots

### Bilingual Rule

\[ X \rightarrow \langle \text{mao, a cat} \rangle \]
\[ X \rightarrow \langle \text{mao, the cat} \rangle \]

### Confusion Rule

\[ X \rightarrow \langle \text{a cat, the cat} \rangle \]
\[ X \rightarrow \langle \text{the cat, a cat} \rangle \]
Step 1: Extracting a Confusion Grammar (CG)

- Deriving a CG from a bilingual grammar

- Use Chinese side as pivots

<table>
<thead>
<tr>
<th>Bilingual Rule</th>
<th>Confusion Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>( X \rightarrow \langle \text{mao, a cat} \rangle )</td>
<td>( X \rightarrow \langle \text{a cat, the cat} \rangle )</td>
</tr>
<tr>
<td>( X \rightarrow \langle \text{mao, the cat} \rangle )</td>
<td>( X \rightarrow \langle \text{the cat, a cat} \rangle )</td>
</tr>
<tr>
<td>( X \rightarrow \langle X_0 \text{ de } X_1, X_0 \text{ on } X_1 \rangle )</td>
<td>( X \rightarrow \langle \text{the cat, a cat} \rangle )</td>
</tr>
<tr>
<td>( X \rightarrow \langle X_0 \text{ de } X_1, X_1 \text{ of } X_0 \rangle )</td>
<td>( X \rightarrow \langle \text{the cat, a cat} \rangle )</td>
</tr>
</tbody>
</table>
**Step-1: Extracting a Confusion Grammar (CG)**

- Deriving a CG from a bilingual grammar
- Use Chinese side as pivots

**Bilingual Rule**

\[
X \rightarrow \langle \text{mao, a cat} \rangle \\
X \rightarrow \langle \text{mao, the cat} \rangle \\
X \rightarrow \langle X_0 \text{ de } X_1, X_0 \text{ on } X_1 \rangle \\
X \rightarrow \langle X_0 \text{ de } X_1, X_1 \text{ of } X_0 \rangle
\]

**Confusion Rule**

\[
X \rightarrow \langle \text{a cat, the cat} \rangle \\
X \rightarrow \langle \text{the cat, a cat} \rangle \\
X \rightarrow \langle X_0 \text{ on } X_1, X_1 \text{ of } X_0 \rangle \\
X \rightarrow \langle X_0 \text{ of } X_1, X_1 \text{ on } X_0 \rangle
\]
Step 1: Extracting a Confusion Grammar (CG)

- Deriving a CG from a bilingual grammar
  - use Chinese side as pivots

<table>
<thead>
<tr>
<th>Bilingual Rule</th>
<th>Confusion Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X \rightarrow \langle \text{mao, a cat} \rangle$</td>
<td>$X \rightarrow \langle \text{a cat, the cat} \rangle$</td>
</tr>
<tr>
<td>$X \rightarrow \langle \text{mao, the cat} \rangle$</td>
<td>$X \rightarrow \langle \text{the cat, a cat} \rangle$</td>
</tr>
<tr>
<td>$X \rightarrow \langle X_0 \text{ de } X_1, X_0 \text{ on } X_1 \rangle$</td>
<td>$X \rightarrow \langle X_0 \text{ on } X_1, X_1 \text{ of } X_0 \rangle$</td>
</tr>
<tr>
<td>$X \rightarrow \langle X_0 \text{ de } X_1, X_1 \text{ of } X_0 \rangle$</td>
<td>$X \rightarrow \langle X_0 \text{ of } X_1, X_1 \text{ on } X_0 \rangle$</td>
</tr>
</tbody>
</table>

CG captures the confusion an MT system will have when translating an input.
**Step-1: Extracting a Confusion Grammar (CG)**

- Deriving a CG from a bilingual grammar
  - use Chinese side as pivots

<table>
<thead>
<tr>
<th>Bilingual Rule</th>
<th>Confusion Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X \rightarrow \langle \text{ mao, a cat} \rangle$</td>
<td>$X \rightarrow \langle \text{ a cat, the cat} \rangle$</td>
</tr>
<tr>
<td>$X \rightarrow \langle \text{ mao, the cat} \rangle$</td>
<td>$X \rightarrow \langle \text{ the cat, a cat} \rangle$</td>
</tr>
<tr>
<td>$X \rightarrow \langle X_0 \text{ de } X_1, X_0 \text{ on } X_1 \rangle$</td>
<td>$X \rightarrow \langle X_0 \text{ on } X_1, X_1 \text{ of } X_0 \rangle$</td>
</tr>
<tr>
<td>$X \rightarrow \langle X_0 \text{ de } X_1, X_1 \text{ of } X_0 \rangle$</td>
<td>$X \rightarrow \langle X_0 \text{ of } X_1, X_1 \text{ on } X_0 \rangle$</td>
</tr>
</tbody>
</table>

CG captures the confusion an MT system will have when translating an input.

Our neighborhood function is learned and MT-specific.
Step-2: Generating Contrastive Sets
Step-2: Generating Contrastive Sets

a cat on the mat
Step-2: Generating Contrastive Sets

a cat on the mat

CG
Step-2: Generating Contrastive Sets

a cat on the mat

CG

S \rightarrow \{X_0, X_0\}

X \rightarrow \{X_0 \text{ on } X_1, X_1 \text{ on } X_0\}

X \rightarrow \{X_0 \text{ on } X_1, X_0 \text{'s } X_1\}

X \rightarrow \{X_0 \text{ on } X_1, X_1 \text{ of } X_0\}

X \rightarrow \{\text{a cat, the cat}\}

X \rightarrow \{\text{the mat, the mat}\}

a_0 \text{ cat}_1 \quad \text{on}_2 \quad \text{the}_3 \text{ mat}_4
Step-2: Generating Contrastive Sets

Given an input sentence "a cat on the mat" the confusion grammar of "a" may generate a hypergraph "i.e.) a contrastive set or neighborhood" for the input sentence) where the hypergraph contains four alternate sentences "the cat the mat") "the cat ’s the mat") "the mat on the cat") and "the mat of the cat".

Formally, let \( S \rightarrow a \) be a confusion grammar. For each non-terminal symbol \( X \) in the grammar, let \( X \rightarrow \{a, b, c\} \) be the set of its productions. The contrastive set \( C(X) \) is defined as:

\[
P(\theta | \tilde{y}_i) = \prod_{d \in D(\tilde{y}_i)} e^{f(d) \cdot \theta}
\]

where \( \theta \) is the contrastive model we aim to train, \( f(d) \) is a feature vector over \( d \), and we will specify in Section 5.4 how the feature functions should be defined. In general, the feature functions should be defined in a way such that the training will be efficient and the actual MT decoding can use them.
Step-2: Generating Contrastive Sets

Contrastive set:

a cat on the mat

the cat the mat
the cat 's the mat
the mat on the cat
the mat of the cat

Given an input sentence “a cat on the mat”) the confusion grammar of “a” may generate a hypergraph “i.e.) a contrastive set or neighborhood” for the input sentence) where the hypergraph contains four alternate sentences “the cat the mat”) “the cat ’s the mat”) “the mat on the cat”) and “the mat of the cat”).

\[
p_\theta(d | \tilde{y}_i) = \frac{e^{f(d)}}{\sum_{d' \in D(\tilde{y}_i)} e^{f(d')}}
\]

where \(\theta\) is the contrastive model we aim to train) and

\[f(d)\]

is a feature vector over \(d\)) for which we will specify in Section 5.4.

In general) the feature functions should be defined in a way such that the training will be efficient and the actual MT decoding can use them.
Step-2: Generating Contrastive Sets

A cat on the mat

Contrastive set:

- The cat the mat
- The cat's the mat
- The mat on the cat
- The mat of the cat

Translating “dianzi shang de mao”? 

a cat on the mat

CG

S \rightarrow \langle X_0, X_0 \rangle

<table>
<thead>
<tr>
<th>X</th>
<th>0, 5</th>
</tr>
</thead>
</table>

\( S \rightarrow \langle X_0, X_0 \rangle \)

\( X \rightarrow \langle X_0 \text{ on } X_1, X_1 \text{ on } X_0 \rangle \)

\( X \rightarrow \langle X_0 \text{ on } X_1, X_0 \text{'s } X_1 \rangle \)

\( X \rightarrow \langle X_0 \text{ on } X_1, X_1 \text{ of } X_0 \rangle \)

\( X \rightarrow \langle \text{a cat, the cat} \rangle \)

\( X \rightarrow \langle \text{the mat, the mat} \rangle \)

\( a_0 \text{ cat} \quad on_2 \quad the_3 \text{ mat} \)

Figure 5.2: Confusion grammar and an example hypergraph generated by the confusion grammar.

Given an input sentence “a cat on the mat”), the confusion grammar of “a” may generate a hypergraph “i.e. a contrastive set or neighborhood” for the input sentence where the hypergraph contains four alternate sentences “the cat the mat”), “the cat's the mat”), “the mat on the cat”), and “the mat of the cat”).

p_\theta(d | \tilde{y}_i) is defined as:

\[ p_\theta(d | \tilde{y}_i) = e^{f(d) \cdot \theta} \]

where \( \theta \) is the contrastive model we aim to train, and \( f(d) \) is a feature vector over \( d \), for which we will specify in Section 5.4. In general, the feature functions should be defined in a way such that the training will be efficient and the actual MT decoding can use them.
Step-3: Discriminative Training
Step-3: Discriminative Training

• Training Objective

\[ \theta^* = \arg \min_{\theta} \sum_{i} \sum_{y \in N(\tilde{y}_i)} L(y, \tilde{y}_i)p_{\theta}(y \mid \tilde{y}_i) \]
Step-3: Discriminative Training

- Training Objective

\[
\theta^* = \arg \min_{\theta} \sum_i \sum_{y \in \mathcal{N}(\tilde{y}_i)} L(y, \tilde{y}_i) p_{\theta}(y | \tilde{y}_i)
\]

\begin{itemize}
  \item \text{contrastive set}
\end{itemize}
Step-3: Discriminative Training

- Training Objective

\[ \theta^* = \arg\min_{\theta} \sum_i \left( \sum_{y \in \mathcal{N}(\tilde{y}_i)} L(y, \tilde{y}_i) p_{\theta}(y | \tilde{y}_i) \right) \]
Step-3: Discriminative Training

- Training Objective

\[ \theta^* = \arg \min_{\theta} \sum \left( \sum_{y \in N(\tilde{y}_i)} L(y, \tilde{y}_i) p_\theta(y \mid \tilde{y}_i) \right) \]

- Expected loss
- Contrastive set
- CE maximizes the conditional likelihood
Step-3: Discriminative Training

• Training Objective

\[ \theta^* = \arg \min_{\theta} \sum \sum_{i} \sum_{y \in \mathcal{N}(\tilde{y}_i)} L(y, \tilde{y}_i)p_{\theta}(y | \tilde{y}_i) \]

Expected loss

Contrastive set

CE maximizes the conditional likelihood

• Iterative Training

  • Step-2: for each English sentence, generate a contrastive set (or neighborhood) using the CG

  • Step-3: discriminative training
Applying the Contrastive Model
Applying the Contrastive Model

• We can use the contrastive model as a regular language model
Applying the Contrastive Model

• We can use the contrastive model as a regular language model

• We can incorporate the contrastive model into an end-to-end MT system as a feature
Applying the Contrastive Model

• We can use the contrastive model as a regular language model

• We can incorporate the contrastive model into an end-to-end MT system as a feature

• We may also use the contrastive model to generate paraphrase sentences (if the loss function measures semantic similarity)

• the rules in CG are symmetric
Test on Synthesized Hypergraphs of English Data
Test on Synthesized Hypergraphs of English Data

Monolingual English

Confusion grammar
Test on Synthesized Hypergraphs of English Data

Monolingual English
Confusion grammar

Training
Test on Synthesized Hypergraphs of English Data

Monolingual English

Confusion grammar

Training

Contrastive LM
Test on Synthesized Hypergraphs of English Data

- Monolingual English
- Confusion grammar
- English Sentence
- Parsing
- Hypergraph (Neighborhood)
- Training
- Contrastive LM
Test on Synthesized Hypergraphs of English Data

Monolingual English

Confusion grammar

Training

Contrastive LM

English Sentence

Parsing

Hypergraph (Neighborhood)

Rank

One-best English
Test on Synthesized Hypergraphs of English Data

Monolingual English

Confusion grammar

Training

Contrastive LM

English Sentence

Parsing

Hypergraph (Neighborhood)

Rank

One-best English

BLEU Score?
Test on Synthesized Hypergraphs of English Data

Monolingual English

Confusion grammar

Training

Contrastive LM

English Sentence

Parsing

Hypergraph (Neighborhood)

Rank

One-best English

BLEU Score?
Results on Synthesized Hypergraphs
Results on Synthesized Hypergraphs

<table>
<thead>
<tr>
<th>BLEU</th>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>26</td>
<td>BLM</td>
</tr>
<tr>
<td>24</td>
<td>+WP</td>
</tr>
<tr>
<td>22</td>
<td>+RuleBigram</td>
</tr>
<tr>
<td>20</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
</tr>
</tbody>
</table>
Results on Synthesized Hypergraphs

Baseline LM (5-gram)
Results on Synthesized Hypergraphs

Features

- baseline LM (5-gram)
- word penalty
- BLM
- +WP
- +RuleBigram

BLEU
Results on Synthesized Hypergraphs

- baseline LM (5-gram)
- word penalty

Features

BLM  +WP  +RuleBigram

BLEU

10  12  14  16  18  20  22  24  26
Results on Synthesized Hypergraphs

```
<table>
<thead>
<tr>
<th>BLEU</th>
<th>BLM</th>
<th>+WP</th>
<th>+RuleBigram</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>12</td>
<td>14</td>
<td>26</td>
</tr>
</tbody>
</table>

baseline LM (5-gram)

word penalty

Features
```
Results on Synthesized Hypergraphs

- **Baseline LM (5-gram)**
- **Word penalty**
- **Features**
  - **BLM**
  - **+WP**
  - **+RuleBigram**

- **Target side of a confusion rule**
  
  “on the $X_1$ issue of $X_2$”
Results on Synthesized Hypergraphs

Baseline LM (5-gram)

Target side of a confusion rule
“on the $X_1$ issue of $X_2$”

Rule bigram features
“on the” “the $X$” “$X$ issue”
“issue of” “of $X$”

Features

BLEU

- BLM
- +WP
- +RuleBigram

Word penalty
The contrastive LM better **recovers** the original English than a regular n-gram LM.

- **Target side of a confusion rule**
  
  “on the $X_1$ issue of $X_2$”

- **Rule bigram features**
  
  “on the”    “the $X$”    “$X$ issue”

  “issue of”    “of $X$”
5.4.4 Results on Monolingual Test Data

We are first interested in seeing how our contrastive language model yCLMfi performs. To directly look at the confusion rule in the confusion grammar ywhich defines the neighbor5, we will use their dominant POS tags yinstead of the symbol including: “and

Adding more features in the CLM helps more in general. These show that our contrastive model performs in picking a good English sentence using

The CLMs ysystem5- to 5’fi that use the CG or Arity feature perform very well ycompared with those systems that do not use themfi7. This is not surprising as these features in the features used during contrastive trainingfi7. yNote that a CLM may include the regular

Language models ythat assign a probability to an English stringfi7. Instead they are models of English strings. Also4 the CG feature is not accessible under the nonterminal of the rule has been removed in the features. Moreover4 for

For example4 if a confusion rule’s target side is “X1, X2” where

- Target side of a confusion rule
  “on the X1 issue of X2”
- Rule bigram features
  “on the” “the X” “X issue” “issue of” “of X”
Results on MT Test Set
Results on MT Test Set

<table>
<thead>
<tr>
<th>BLEU</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
</tr>
<tr>
<td>49</td>
</tr>
<tr>
<td>48</td>
</tr>
<tr>
<td>47</td>
</tr>
<tr>
<td>46</td>
</tr>
<tr>
<td>45</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Features</th>
<th>Baseline</th>
<th>+CLM</th>
</tr>
</thead>
</table>

Results on MT Test Set

[Diagram showing BLEU scores for Baseline and +CLM features]

- **Features**
  - Bilingual Data
  - Monolingual English

- **Training**
  - Generative Training
  - Discriminative Training

- **Models**
  - Translation Models
  - Language Models

- **Data**
  - Held-out Bilingual Data

- **Outputs**
  - Translation Outputs

- **Processes**
  - Unseen Sentences
  - Decoding

- **Optimal Weights**
Results on MT Test Set

- **Features**
  - Bilingual Data
  - Monolingual English

- **Generative Training**
  - Translation Models
  - Language Models

- **Discriminative Training**
  - Optimal Weights

- **Held-out Bilingual Data**

- **Decoding**
  - Translation Outputs

Add CLM as a feature
The contrastive LM helps to improve MT performance.
Adding Features on the CG itself

- On English Set

- On MT Set
Adding Features on the CG itself

- On English Set

- On MT Set
Adding Features on the CG itself

- On English Set

- On MT Set
Adding Features on the CG itself

- **On English Set**

- **On MT Set**
Adding Features on the CG itself

• On English Set

• On MT Set
Adding Features on the CG itself

- On English Set

- On MT Set

![BLEU Chart](image)

Features

<table>
<thead>
<tr>
<th>BLEU</th>
<th>BLM</th>
<th>+WP</th>
<th>+RuleBigram</th>
<th>+Arity</th>
<th>+CG</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>35</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>40</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>45</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

glue rules or regular confusion rules?

\[ S \rightarrow \langle S_0 X_1, S_0 X_1 \rangle \]

\[ S \rightarrow \langle X_0, X_0 \rangle \]
Adding Features on the CG itself

- On English Set

- On MT Set

- BLEU

- Features

- One big feature

- glue rules or regular confusion rules?

- $S \rightarrow \langle S_0 X_1, S_0 X_1 \rangle$

- $S \rightarrow \langle X_0, X_0 \rangle$
Adding Features on the CG itself

- On English Set

- On MT Set

\[ S \rightarrow \langle S_0 X_1, S_0 X_1 \rangle \]

\[ S \rightarrow \langle X_0, X_0 \rangle \]
Adding Features on the CG itself

- On English Set

- On MT Set

\[ S \rightarrow \langle S_0, X_1, S_0, X_1 \rangle \]
\[ S \rightarrow \langle X_0, X_0 \rangle \]
Adding Features on the CG itself

• On English Set

![Graph showing BLEU scores for different configurations: BLM, +WP, +RuleBigram, +Arity, +CG.](image)

- **Paraphrasing model**
- **One big feature**
- **Glue rules or regular confusion rules?**

  $S \rightarrow \langle S_0 X_1, S_0 X_1 \rangle$

  $S \rightarrow \langle X_0, X_0 \rangle$

• On MT Set
Adding Features on the CG itself

- **On English Set**

- **On MT Set**

![Graph showing BLEU scores for different features and datasets]
Adding Features on the CG itself

- On English Set

- On MT Set

Features

Paraphrasing model

one big feature

going rules or regular confusion rules?

\[ S \rightarrow \langle S_0, X_1, S_0, X_1 \rangle \]

\[ S \rightarrow \langle X_0, X_0 \rangle \]
Summary for Discriminative Training

- **Supervised:** Minimum Empirical Risk
  
  
  require bitext

  \[
  x \xrightarrow{\delta_\theta} \delta_\theta(x) \rightleftharpoons y
  \]

- **Unsupervised:** Minimum Imputed Risk
  
  require monolingual English

  \[
  x \xleftarrow{p_\phi} \xrightarrow{\delta_\theta} \delta_\theta(x) \xrightarrow{\tilde{y}_i} \tilde{y}_i \xleftarrow{\text{loss}}
  \]

- **Unsupervised:** Contrastive LM Estimation
  
  require monolingual English

  \[
  \tilde{y} \xrightarrow{\text{loss}} \text{Neighborhood Function} \xrightarrow{\mathcal{N}(\tilde{y})}
  \]
Summary for Discriminative Training

• **Supervised:** Minimum Empirical Risk

  ![Diagram](image)

  - Require bitext
  - Parameterized by $\phi$
  - Objective: $\min_{\theta} \mathbb{E}_{(x,y) \sim D} L(x, \delta_\theta(x), \delta_\theta(y))$

• **Unsupervised:** Minimum Imputed Risk

  ![Diagram](image)

  - Require monolingual English
  - Objective: $\min_{\theta, \phi} \mathbb{E}_{x \sim D} \mathbb{E}_{y \sim p(y|x)} L(x, \delta_\theta(x), \tilde{y}_i)$

• **Unsupervised:** Contrastive LM Estimation

  ![Diagram](image)

  - Require monolingual English
  - Objective: $\min_{\theta} \mathbb{E}_{y \sim D} \mathbb{E}_{\tilde{y} \sim \mathcal{N}(y)} L(y, \tilde{y})$

  - Neighborhood Function $\mathcal{N}$

---

Learn a good understanding of the corresponding input data. A method for discriminative training is described in this chapter. Our method is theoretically sound and can be explained as minimizing imputed risk. It is also intuitive: it tries to ensure that probabilistic “round-trip” translation from the target language sentence to the source language and back again will have low expected loss. Our method works as follows: first, guess a reverse prediction model that attempts to impute the missing data. We will train this model for discriminative training. Our method may be used for other tasks as well. For example, in a speech recognition task, what we called “audio 2” or at least over acoustic features or phone sequences.

Our method for discriminative training: 

1. Guess a reverse prediction model $p_\phi$ that attempts to impute the missing data.
2. Train this model for discriminative training.

We perform experiments by using the open-source MT toolkit. Our goal is to find the best job we can from available data—including our bilingual data—and choose the one that optimizes the given performance metric. The trouble is that a typical reverse model may have any form and need not be probabilistic. For example, the parameters of some complex translation system are tuned by pruning and decoding heuristics for extracting a high-scoring translation. Now do ordinary supervised training as of 2.

Specific to an MT task, our method works as follows: first, guess a reverse prediction model $p_\phi$ that attempts to impute the missing data. We will train this model for discriminative training. Our method may be used for other tasks as well. For example, in a speech recognition task, what we called “audio 2” or at least over acoustic features or phone sequences.

One wishes to tune the parameters of some complex translation system by minimizing imputed risk. For example, the parameters of some complex translation system are tuned by pruning and decoding heuristics for extracting a high-scoring translation. Now do ordinary supervised training as of 2.

4. Is 0 by tuning $\theta$.

4. is $\min_{\theta} \mathbb{E}_{i \sim D} L_{i, x}^{i, x}$ where $X$ is a kind of speech synthesizer that must produce a distribution over $y$.

5. It is computationally infeasible to compute $\min_{\theta} \mathbb{E}_{(x,y) \sim D} L(x, \delta_\theta(x), \delta_\theta(y))$ directly.

We approximate this by computing $\min_{\theta, \phi} \mathbb{E}_{x \sim D} \mathbb{E}_{y \sim p(y|x)} L(x, \delta_\theta(x), \tilde{y}_i)$, where $\tilde{y}_i$ is a weighted lattice or hypergraph of some complex translation system.

We will train this model for discriminative training. Our goal is to find the best job we can from available data—including our bilingual data—and choose the one that optimizes the given performance metric. The trouble is that a typical reverse model may have any form and need not be probabilistic. For example, the parameters of some complex translation system are tuned by pruning and decoding heuristics for extracting a high-scoring translation. Now do ordinary supervised training as of 2.

Specific to an MT task, our method works as follows: first, guess a reverse prediction model $p_\phi$ that attempts to impute the missing data. We will train this model for discriminative training. Our method may be used for other tasks as well. For example, in a speech recognition task, what we called “audio 2” or at least over acoustic features or phone sequences.

One wishes to tune the parameters of some complex translation system by minimizing imputed risk. For example, the parameters of some complex translation system are tuned by pruning and decoding heuristics for extracting a high-scoring translation. Now do ordinary supervised training as of 2.
Summary for Discriminative Training

- **Supervised Training**
  - Require bitext

- **Unsupervised: Minimum Imputed Risk**
  - Require monolingual English

- **Unsupervised: Contrastive LM Estimation**
  - Require monolingual English
Summary for Discriminative Training

- **Supervised Training**

  Require bitext

  \[
  x \xrightarrow{\delta_\theta} \delta_\theta(x) \xrightarrow{\text{loss}} y
  \]

- **Unsupervised: Minimum Imputed Risk**

  Require monolingual English

  \[
  x \xleftarrow{\mathcal{P}_\phi} \tilde{y}_i \xrightarrow{\text{loss}} \delta_\theta(x) \xrightarrow{\text{loss}} \delta_\theta(x)
  \]

  Require a reverse model

- **Unsupervised: Contrastive LM Estimation**

  Require monolingual English

  \[
  \tilde{y} \xrightarrow{\text{loss}} \text{Neighborhood Function} \xrightarrow{\mathcal{N}(\tilde{y})}
  \]
Summary for Discriminative Training

- **Supervised Training**
  
  \[
  x \rightarrow \delta_\theta \rightarrow \delta_\theta(x) \rightarrow y
  \]
  
  require bitext

- **Unsupervised: Minimum Imputed Risk**
  
  \[
  x \rightarrow P_\phi \rightarrow \tilde{y}_i \rightarrow \delta_\theta \rightarrow \delta_\theta(x)
  \]
  
  require a reverse model
  can have both TM and LM features

- **Unsupervised: Contrastive LM Estimation**
  
  \[
  \tilde{y} \rightarrow \text{Neighborhood Function} \rightarrow \mathcal{N}(\tilde{y})
  \]
  
  require monolingual English

One wishes to tune the parameters in the sense of optimizing the given performance metric. Our method is also intuitive: it tries to ensure that probabilistic "round-trip" translation from the target-language sentence to forward translate, then back again, will have a low "expected" loss.

In this chapter, we describe an conditional model which translates Chinese to English system to do a "good job" at translating this imputed native Chinese to English system. Also, adding unsupervised data into the supervised training often improves the performance. Our experiments show that unsupervised discriminative training performs similarly to the supervised case.

Specific to an MT task, our method works as follows. First, guess a "reverse prediction model" that attempts to impute the missing native Chinese sentences. Then, train the discriminator using a reverse English to Chinese model. Also, assume that we have a minimum imputed-risk parameterized by \( \theta \), and choose the \( \theta \) that minimizes the weighted sum of these losses, i.e., the empirical risk when the empirical result of \( \delta_\theta(x) \) is derived from the imputed training set. Specific to our MT task, note that \( \delta_\theta \) is a kind of speech synthesizer that must produce a distribution over parameterized by \( \phi \). We can also exploit both supervised and unsupervised data to perform semi-supervised training by using reverse translations.
Summary for Discriminative Training

- **Supervised Training**

  require bitext

  \[ x \rightarrow \delta_\theta \rightarrow \delta_\theta(x) \rightarrow y \]

- **Unsupervised: Minimum Imputed Risk**

  require a reverse model

  can have both TM and LM features

  \[ x \leftarrow P_\phi \leftarrow \tilde{y}_i \]

  \[ \delta_\theta \rightarrow \delta_\theta(x) \]

- **Unsupervised: Contrastive LM Estimation**

  can have LM features only

  require monolingual English

  \[ \tilde{y} \rightarrow \text{Neighborhood Function} \rightarrow \mathcal{N}(\tilde{y}) \]

Specific to an MT task, our method works as follows. First, guess the corresponding input data. The second step means that we must use pruning and decoding heuristics for extracting a high-scoring translation. For each unsupervised example, we can use the reverse model (a "reverse prediction model" that attempts to impute the missing data). We will train this to forward, translate, and reverse. The crucial ingredient here is the minimum imputed-risk objective of \( \min_{\theta} \mathbb{E}_{(x,y) \sim D} [L_{\text{impute}}(\delta_\theta(x), \tilde{y})] \), where \( \mathbb{E} \) denotes the expected loss. Our method is theoretically sound and can be explained as minimizing imputed risk. It is a "good job" at translating this imputed function. In this chapter, we describe an open-source MT toolkit. We perform experiments by using the open-source MT toolkit. Our method is also intuitive: it tries to ensure that probabilistic "roundtrip" translation reasonably good.
Summary for Discriminative Training

- **Supervised Training**
  - require bitext
  - \[ x \rightarrow \delta_{\theta} \rightarrow \delta_{\theta}(x) \rightarrow \text{loss} \rightarrow y \]

- **Unsupervised: Minimum Imputed Risk**
  - require monolingual English
  - require a reverse model
  - can have both TM and LM features

- **Unsupervised: Contrastive LM Estimation**
  - require monolingual English
  - can have LM features only
Outline

- Hypergraph as Hypothesis Space
- Unsupervised Discriminative Training
  - minimum imputed risk
  - contrastive language model estimation
- Variational Decoding
- First- and Second-order Expectation Semirings

<table>
<thead>
<tr>
<th>decoding</th>
<th>training</th>
</tr>
</thead>
<tbody>
<tr>
<td>(e.g., mbr)</td>
<td>(e.g., mert)</td>
</tr>
</tbody>
</table>

atomic inference operations
(e.g., finding one-best, k-best or expectation, inference can be exact or approximate)
Variational Decoding
Variational Decoding

• We want to do inference under $p$, but it is intractable
Variational Decoding

• We want to do inference under $p$, but it is intractable.

• Instead, we derive a simpler distribution $q^*$.
Variational Decoding

• We want to do inference under $p$, but it is intractable

• Instead, we derive a simpler distribution $q^*$

• Then, we will use $q^*$ as a surrogate for $p$ in inference
Variational Decoding

• We want to do inference under $p$, but it is intractable.

intractable MAP decoding

• Instead, we derive a simpler distribution $q^*$

• Then, we will use $q^*$ as a surrogate for $p$ in inference
Variational Decoding

• We want to do inference under $p$, but it is intractable

\[ y^* = \arg \max_y p(y | x) \]

• Instead, we derive a simpler distribution $q^*$

• Then, we will use $q^*$ as a surrogate for $p$ in inference
Variational Decoding

• We want to do inference under $p$, but it is intractable

intractable MAP decoding

$$y^* = \arg \max_y p(y | x) = \arg \max_y \sum_{d \in D(x,y)} p(d | x)$$

• Instead, we derive a simpler distribution $q^*$

• Then, we will use $q^*$ as a surrogate for $p$ in inference
Variational Decoding

- We want to do inference under $p$, but it is intractable.

**Intractable MAP decoding** (Sima'an 1996)

$$
y^* = \arg \max_y p(y \mid x) = \arg \max_y \sum_{d \in D(x,y)} p(d \mid x)
$$

- Instead, we derive a simpler distribution $q^*$

- Then, we will use $q^*$ as a surrogate for $p$ in inference
Variational Decoding

• We want to do inference under $p$, but it is intractable

\[
y^* = \operatorname{arg\,max}_y p(y \mid x) = \operatorname{arg\,max}_y \sum_{d \in D(x, y)} p(d \mid x)
\]

• Instead, we derive a simpler distribution $q^*$

\[
q^* = \operatorname{arg\,min}_{q \in Q} \text{KL}(p \parallel q)
\]

• Then, we will use $q^*$ as a surrogate for $p$ in inference
Variational Decoding

• We want to do inference under $p$, but it is intractable

\[ y^* = \arg \max_y p(y \mid x) = \arg \max_y \sum_{d \in D(x,y)} p(d \mid x) \]

• Instead, we derive a simpler distribution $q^*$

**tractable estimation**

\[ q^* = \arg \min_{q \in Q} \text{KL}(p \mid \mid q) \]

• Then, we will use $q^*$ as a surrogate for $p$ in inference
Variational Decoding

- We want to do inference under \( p \), but it is intractable intractable MAP decoding (Sima’an 1996)

\[
y^* = \arg\max_y p(y \mid x) = \arg\max_y \sum_{d \in D(x,y)} p(d \mid x)
\]

- Instead, we derive a simpler distribution \( q^* \)

tractable estimation

\[
q^* = \arg\min_{q \in Q} KL(p\|q)
\]

- Then, we will use \( q^* \) as a surrogate for \( p \) in inference
Variational Decoding

- We want to do inference under $p$, but it is intractable

**intractable MAP decoding** (Sima’an 1996)

$$y^* = \arg\max_y p(y \mid x) = \arg\max_y \sum_{d \in \mathcal{D}(x,y)} p(d \mid x)$$

- Instead, we derive a simpler distribution $q^*$

**tractable estimation**

$$q^* = \arg\min_{q \in \mathcal{Q}} \text{KL}(p \parallel q)$$

- Then, we will use $q^*$ as a surrogate for $p$ in inference
Variational Decoding

- We want to do inference under $p$, but it is intractable.

**intractable MAP decoding** (Sima’an 1996)

$$ y^* = \arg\max_y p(y \mid x) = \arg\max_y \sum_{d \in D(x,y)} p(d \mid x) $$

- Instead, we derive a simpler distribution $q^*$

**tractable estimation**

$$ q^* = \arg\min_{q \in Q} KL(p \| q) $$

- Then, we will use $q^*$ as a surrogate for $p$ in inference.
Variational Decoding

• We want to do inference under $p$, but it is intractable.

**intractable MAP decoding** (Sima’an 1996)

$$y^* = \arg\max_y p(y \mid x) = \arg\max_y \sum_{d \in D(x, y)} p(d \mid x)$$

• Instead, we derive a simpler distribution $q^*$

**tractable estimation**

$$q^* = \arg\min_{q \in Q} KL(p \| q)$$

• Then, we will use $q^*$ as a surrogate for $p$ in inference.

**tractable decoding**

$$y^* = \arg\max_y q^*(y \mid x)$$
Variational Decoding for MT: an Overview
Variational Decoding for MT: an Overview

Sentence-specific decoding
Variational Decoding for MT: an Overview

Sentence-specific decoding

Three steps:
Variational Decoding for MT: an Overview

Sentence-specific decoding

Three steps:

1. Generate a hypergraph for the foreign sentence
Variational Decoding for MT: an Overview

Sentence-specific decoding

Three steps:

1. Generate a hypergraph for the foreign sentence

Foreign sentence $x$
Variational Decoding for MT: an Overview

Sentence-specific decoding

Three steps:

1. Generate a hypergraph for the foreign sentence

MAP decoding under P is intractable

Foreign sentence $x$ → SMT → $p(d \mid x)$

$p(y \mid x) = \sum_{d \in D(x, y)} p(d \mid x)$

$p(y \mid x) = \sum_{d \in D(x, y)} p(d \mid x)$
Generate a hypergraph

\[ p(d \mid x) \]
Generate a hypergraph
Generate a hypergraph
Generate a hypergraph
Generate a hypergraph

Estimate a model from the hypergraph by minimizing KL

\[ q^* \text{ is an n-gram model over output strings.} \]
1. Generate a hypergraph

2. Estimate a model from the hypergraph by minimizing KL

\[ q^*(y | x) \approx \sum_{d \in D(x, y)} p(d | x) \]

\( q^* \) is an n-gram model over output strings.
Generate a hypergraph

Estimate a model from the hypergraph by minimizing KL

q* is an n-gram model over output strings.

\[ q^*(y \mid x) \approx \sum_{d \in D(x, y)} p(d \mid x) \]
Generate a hypergraph

Estimate a model from the hypergraph by minimizing KL

Decode using $q^*$ on the hypergraph

$q^*(y | x) \approx \sum_{d \in D(x,y)} p(d|x)$

$q^*$ is an n-gram model over output strings.
Given a hypergraph $p(d | x)$, we can:

1. **Generate a hypergraph**
2. Estimate a model from the hypergraph by minimizing KL divergence...
3. Approximate a hypergraph with a lattice...
4. Decode using $q^*$ on the hypergraph

$p(d | x)$ is an n-gram model over output strings.

$$q^*(y | x) \approx \sum_{d \in D(x,y)} p(d|x)$$
Outline

- Hypergraph as Hypothesis Space
- Unsupervised Discriminative Training
  - minimum imputed risk
  - contrastive language model estimation
- Variational Decoding
- First- and Second-order Expectation Semirings

<table>
<thead>
<tr>
<th>decoding</th>
<th>training</th>
</tr>
</thead>
<tbody>
<tr>
<td>(e.g., mbr)</td>
<td>(e.g., mert)</td>
</tr>
</tbody>
</table>

atomic inference operations
(e.g., finding one-best, k-best or expectation, inference can be exact or approximate)
A semiring framework to compute all of these

- **First-order expectations:**
  - expectation
  - entropy
  - expected loss
  - cross-entropy
  - KL divergence
  - feature expectations
  - first-order gradient of $Z$

- **Second-order expectations:**
  - expectation over product
  - interaction between features
  - Hessian matrix of $Z$
  - second-order gradient descent
  - gradient of expectation
  - gradient of expected loss or entropy

- “Decoding” quantities:
  - Viterbi
  - K-best
  - Counting
  - ......
A semiring framework to compute all of these

**First-order expectations:**
- expectation
- entropy
- expected loss
- cross-entropy
- KL divergence
- feature expectations
- first-order gradient of $Z$

**Second-order expectations:**
- expectation over product
- interaction between features
- Hessian matrix of $Z$
- second-order gradient descent
- gradient of expectation
- gradient of expected loss or entropy

**“Decoding” quantities:**
- Viterbi
- K-best
- Counting
- ......
A semiring framework to compute all of these quantities:

- Expectation
- Entropy
- Expected loss
- Cross-entropy
- KL divergence
- Feature expectations
- First-order gradient of $Z$
- Interaction between features
- Hessian matrix of $Z$
- Second-order gradient descent
- Gradient of expectation
- Gradient of expected loss or entropy

**Recipe to compute a quantity:**
A semiring framework to compute all of these

Recipe to compute a quantity:

- Choose a semiring

First-order expectations:
- expectation
- entropy
- expected loss
- cross-entropy
- KL divergence
- feature expectations
- first-order gradient of $Z$

Second-order expectations:
- expectation over product
- interaction between features
- Hessian matrix of $Z$
- second-order gradient descent
- gradient of expectation
- gradient of expected loss or entropy

"Decoding" quantities:
- Viterbi
- K-best
- Counting
A semiring framework to compute all of these

**First-order expectations:**
- expectation
- entropy
- expected loss
- cross-entropy
- KL divergence
- feature expectations
- first-order gradient of $Z$

**Second-order expectations:**
- expectation over product
- interaction between features
- Hessian matrix of $Z$
- second-order gradient descent
- gradient of expectation
- gradient of expected loss or entropy

**“Decoding” quantities:**
- Viterbi
- K-best
- Counting

**Recipe to compute a quantity:**

- Choose a semiring
- Specific a semiring weight for each hyperedge
A semiring framework to compute all of these

• First-order expectations:
  - expectation
  - entropy
  - expected loss
  - cross-entropy
  - KL divergence
  - feature expectations
  - first-order gradient of \( Z \)

• Second-order expectations:
  - expectation over product
  - interaction between features
  - Hessian matrix of \( Z \)
  - second-order gradient descent
  - gradient of expectation
  - gradient of expected loss or entropy

Recipe to compute a quantity:

• Choose a semiring
• Specific a semiring weight for each hyperedge
• Run the inside algorithm

Probabilistic Hypergraph

"Decoding" quantities:

- Viterbi
- K-best
- Counting

\( X_{0,4} \) the cat
\( X_{0,4} \) a mat
\( \text{dianzi} \) shang
\( X_{0,2} \) the mat
\( X_{(\text{dianzi} \text{ shang} \text{ the mat})} \)
\( X_{0} \) de \( X_{1} \)
\( X_{0} \) on \( X_{0} \)
\( X_{0} \) of \( X_{0} \)
\( S \rightarrow \) X_{0}
Applications of Expectation Semirings: a Summary

<table>
<thead>
<tr>
<th>Quantity</th>
<th>$k_e$</th>
<th>$k_{\text{root}}$</th>
<th>Final</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expectation</td>
<td>$\langle p_e, p_e r_e \rangle$</td>
<td>$\langle Z, \bar{r} \rangle$</td>
<td>$\bar{r}/Z$</td>
</tr>
<tr>
<td>Entropy</td>
<td>$r_e \overset{\text{def}}{=} \log p_e$, so $k_e = \langle p_e, p_e \log p_e \rangle$</td>
<td>$\langle Z, \bar{r} \rangle$</td>
<td>$\log Z - \bar{r}/Z$</td>
</tr>
<tr>
<td>Cross-entropy</td>
<td>$r_e \overset{\text{def}}{=} \log q_e$, so $k_e = \langle p_e, p_e \log q_e \rangle$</td>
<td>$\langle Z_p, \bar{r} \rangle$</td>
<td>$\log Z_q - \bar{r}/Z_p$</td>
</tr>
<tr>
<td>Bayes risk</td>
<td>$r_e \overset{\text{def}}{=} L_e$, so $k_e = \langle p_e, p_e L_e \rangle$</td>
<td>$\langle Z, \bar{r} \rangle$</td>
<td>$\bar{r}/Z$</td>
</tr>
<tr>
<td>First-order gradient</td>
<td>$\langle p_e, \nabla p_e \rangle$</td>
<td>$\langle Z, \nabla Z \rangle$</td>
<td>$\nabla Z$</td>
</tr>
<tr>
<td>Covariance matrix</td>
<td>$\langle p_e, p_e r_e, p_e s_e, p_e r_e s_e \rangle$</td>
<td>$\langle Z, \bar{r}, \bar{s}, \bar{t} \rangle$</td>
<td>$\frac{\bar{t}}{Z} - \frac{\bar{r} \bar{s}^T}{Z^2}$</td>
</tr>
<tr>
<td>Hessian matrix</td>
<td>$\langle p_e, \nabla p_e, \nabla p_e, \nabla^2 p_e \rangle$</td>
<td>$\langle Z, \nabla Z, \nabla Z, \nabla^2 Z \rangle$</td>
<td>$\nabla^2 Z$</td>
</tr>
<tr>
<td>Gradient of expectation</td>
<td>$\langle p_e, p_e r_e, \nabla p_e, (\nabla p_e) r_e + p_e (\nabla r_e) \rangle$</td>
<td>$\langle Z, \bar{r}, \nabla Z, \nabla \bar{r} \rangle$</td>
<td>$\frac{Z \nabla \bar{r} - \bar{r} \nabla Z}{Z^2}$</td>
</tr>
<tr>
<td>Gradient of entropy</td>
<td>$\langle p_e, p_e \log p_e, \nabla p_e, (1 + \log p_e) \nabla p_e \rangle$</td>
<td>$\langle Z, \bar{r}, \nabla Z, \nabla \bar{r} \rangle$</td>
<td>$\frac{\nabla Z}{Z} - \frac{Z \nabla \bar{r} - \bar{r} \nabla Z}{Z^2}$</td>
</tr>
<tr>
<td>Gradient of risk</td>
<td>$\langle p_e, p_e L_e, \nabla p_e, L_e \nabla p_e \rangle$</td>
<td>$\langle Z, \bar{r}, \nabla Z, \nabla \bar{r} \rangle$</td>
<td>$\frac{Z \nabla \bar{r} - \bar{r} \nabla Z}{Z^2}$</td>
</tr>
</tbody>
</table>
Inference, Training and Decoding on Hypergraphs

- Unsupervised Discriminative Training
  - minimum imputed risk \((\text{In Preparation})\)
  - contrastive language model estimation \((\text{In Preparation})\)

- Variational Decoding
  \((\text{Li et al., ACL 2009})\)

- First- and Second-order Expectation Semirings
  \((\text{Li and Eisner, EMNLP 2009})\)
My Other MT Research

• **Training methods (supervised)**
  - Discriminative forest reranking with Perceptron  
    (Li and Khudanpur, GALE book chapter 2009)
  - Discriminative n-gram language models  
    (Li and Khudanpur, AMTA 2008)

• **Algorithms**
  - Oracle extraction from hypergraphs  
    (Li and Khudanpur, NAACL 2009)
  - Efficient intersection between n-gram LM and CFG  
    (Li and Khudanpur, ACL SSST 2008)

• **Others**
  - System combination (Smith et al., GALE book chapter 2009)
  - Unsupervised translation induction for Chinese abbreviations  
    (Li and Yarowsky, ACL 2008)
• **Information extraction**
  • Relation extraction between formal and informal phrases (Li and Yarowsky, EMNLP 2008)

• **Spoken dialog management**
  • Optimal dialog in consumer-rating systems using a POMDP (Li et al., SIGDial 2008)
Joshua project

• An open-source parsing-based MT toolkit (Li et al. 2009)
  • support Hiero (Chiang, 2007) and SAMT (Venugopal et al., 2007)

• Team members
  • Zhifei Li, Chris Callison-Burch, Chris Dyer, Sanjeev Khudanpur, Wren Thornton, Jonathan Weese, Juri Ganitkevitch, Lane Schwartz, and Omar Zaidan

Only rely on word-aligner and SRI LM!
All the methods presented have been implemented in Joshua!
Thank you!

XieXie!

谢谢!
Decoding over a hypergraph
Decoding over a hypergraph

Given a hypergraph of possible translations
(generated for a given foreign sentence by already-trained model)
Decoding over a hypergraph

*Given a hypergraph of possible translations* (generated for a given foreign sentence by already-trained model)

Pick a single translation to output
(why not just pick the tree with the highest weight?)
Spurious Ambiguity

• Statistical models in MT exhibit spurious ambiguity
  • Many different derivations (e.g., trees or segmentations) generate the same translation string

• Tree-based MT systems
  • derivation tree ambiguity

• Regular phrase-based MT systems
  • phrase segmentation ambiguity
Spurious Ambiguity in Derivation Trees
Spurious Ambiguity in Derivation Trees

machine translation software

jiqi fanyi yuanjian
Spurious Ambiguity in Derivation Trees

jiqi fanyi yuanjian

S -> (机器, machine)  S -> (翻译, translation)  S -> (软件, software)

machine translation software
Spurious Ambiguity in Derivation Trees

jiqi fanyi yuanjian

machine translation software
Spurious Ambiguity in Derivation Trees

jiqi fanyi yuanjian

S->(S0 S1, S0 S1)

S->(机器, machine)  S->(翻译, translation)  S->(软件, software)

S->(S0 S1, S0 S1)

S->(机器, machine)  S->(翻译, translation)  S->(软件, software)

S->(S0 翻译 S1, S0 translation S1)

S->(机器, machine)  翻译  S->(软件, software)
Spurious Ambiguity in Derivation Trees

Same output: “machine translation software”

Three different derivation trees

jiqi fanyi yuanjian

machine translation software
Spurious Ambiguity in Derivation Trees

Same output: “machine translation software”

Three different derivation trees

Another translation: machine transfer software

jiqi fanyi yuanjian
MAP, Viterbi and N-best Approximations
MAP, Viterbi and N-best Approximations

- Exact MAP decoding

\[ y^* = \arg \max_{y \in \text{Trans}(x)} p(y|x) \]

\[ = \arg \max_{y \in \text{Trans}(x)} \sum_{d \in \mathcal{D}(x,y)} p(y, d|x) \]
MAP, Viterbi and N-best Approximations

• Exact MAP decoding

\[ y^* = \arg\max_{y \in \text{Trans}(x)} p(y|x) \]

\[ = \arg\max_{y \in \text{Trans}(x)} \sum_{d \in \text{D}(x,y)} p(y, d|x) \]

NP-hard (Sima’an 1996)
MAP, Viterbi and N-best Approximations

- **Exact MAP decoding**

\[
y^* = \arg \max_{y \in \text{Trans}(x)} p(y|x)
\]

\[
y^* = \arg \max_{y \in \text{Trans}(x)} \sum_{d \in \text{D}(x,y)} p(y, d|x)\]

NP-hard (Sima’an 1996)

- **Viterbi approximation**

\[
y^* = \arg \max_{y \in \text{Trans}(x)} \max_{d \in \text{D}(x,y)} p(y, d|x)
\]
MAP, Viterbi and N-best Approximations

• Exact MAP decoding
\[
y^* = \arg \max_{y \in \text{Trans}(x)} p(y|x)
\]
\[
= \arg \max_{y \in \text{Trans}(x)} \sum_{d \in D(x,y)} p(y, d|x)
\]
NP-hard (Sima’an 1996)

• Viterbi approximation
\[
y^* = \arg \max_{y \in \text{Trans}(x)} \max_{d \in D(x,y)} p(y, d|x)
\]
\[
= Y(\arg \max_{d \in D(x)} p(y, d|x))
\]
MAP, Viterbi and N-best Approximations

• Exact MAP decoding

\[
y^{*} = \arg \max_{y \in \text{Trans}(x)} p(y|\mathbf{x}) = \arg \max_{y \in \text{Trans}(x)} \sum_{d \in \mathcal{D}(x,y)} p(y, d|\mathbf{x})
\]

NP-hard (Sima’an 1996)

• Viterbi approximation

\[
y^{*} = \arg \max_{y \in \text{Trans}(x)} \max_{d \in \mathcal{D}(x,y)} p(y, d|\mathbf{x}) = \mathcal{Y}(\arg \max_{d \in \mathcal{D}(x)} p(y, d|\mathbf{x}))
\]
MAP, Viterbi and N-best Approximations

- **Exact MAP decoding**

\[
y^* = \arg \max_{y \in \text{Trans}(x)} p(y|x)
\]

\[
y^* = \arg \max_{y \in \text{Trans}(x)} \sum_{d \in \text{D}(x,y)} p(y, d|x)
\]

NP-hard (Sima’an 1996)

- **Viterbi approximation**

\[
y^* = \arg \max_{y \in \text{Trans}(x)} \max_{d \in \text{D}(x,y)} p(y, d|x)
\]

\[
y^* = \mathcal{Y}(\arg \max_{d \in \text{D}(x)} p(y, d|x))
\]
MAP, Viterbi and N-best Approximations

- **Exact MAP decoding**
  
  \[
  y^* = \arg \max_{y \in \text{Trans}(x)} p(y|x) \quad \text{NP-hard (Sima’an 1996)}
  \]
  
  \[
  = \arg \max_{y \in \text{Trans}(x)} \sum_{d \in D(x,y)} p(y, d|x)
  \]

- **Viterbi approximation**
  
  \[
  y^* = \arg \max_{y \in \text{Trans}(x)} \max_{d \in D(x,y)} p(y, d|x)
  \]
  
  \[
  = Y(\arg \max_{d \in D(x)} p(y, d|x))
  \]

- **N-best approximation (crunching) (May and Knight 2006)**
  
  \[
  y^* = \arg \max_{y \in \text{Trans}(x)} \sum_{d \in D(x,y) \cap \text{ND}(x)} p(y, d|x)
  \]
MAP, Viterbi and N-best Approximations

- **Exact MAP decoding**
  \[
  y^* = \arg \max_{y \in \text{Trans}(x)} p(y|x) \quad \text{NP-hard (Sima’an 1996)}
  \]
  \[
  = \arg \max_{y \in \text{Trans}(x)} \sum_{d \in \mathcal{D}(x, y)} p(y, d|x)
  \]

- **Viterbi approximation**
  \[
  y^* = \arg \max_{y \in \text{Trans}(x)} \max_{d \in \mathcal{D}(x, y)} p(y, d|x)
  \]
  \[
  = \text{Y}(\arg \max_{d \in \mathcal{D}(x)} p(y, d|x))
  \]

- **N-best approximation (crunching) (May and Knight 2006)**
  \[
  y^* = \arg \max_{y \in \text{Trans}(x)} \sum_{d \in \mathcal{D}(x, y) \cap \text{ND}(x)} p(y, d|x)
  \]

66
MAP, Viterbi and N-best Approximations

- **Exact MAP decoding**
  \[
  y^* = \arg \max_{y \in \text{Trans}(x)} p(y|x) \quad \text{NP-hard (Sima’an 1996)}
  \]
  \[
  = \arg \max_{y \in \text{Trans}(x)} \sum_{d \in \text{D}(x,y)} p(y, d|x)
  \]

- **Viterbi approximation**
  \[
  y^* = \arg \max_{y \in \text{Trans}(x)} \max_{d \in \text{D}(x,y)} p(y, d|x)
  \]
  \[
  = \text{Y}(\arg \max_{d \in \text{D}(x)} p(y, d|x))
  \]

- **N-best approximation (crunching) (May and Knight 2006)**
  \[
  y^* = \arg \max_{y \in \text{Trans}(x)} \sum_{d \in \text{D}(x,y) \cap \text{ND}(x)} p(y, d|x)
  \]
# MAP vs. Approximations

<table>
<thead>
<tr>
<th>translation string</th>
<th>MAP</th>
<th>Viterbi</th>
<th>4-best crunching</th>
<th>derivation</th>
<th>probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>red translation</td>
<td>0.28</td>
<td>0.16</td>
<td>0.16</td>
<td></td>
<td>0.16</td>
</tr>
<tr>
<td>blue translation</td>
<td>0.28</td>
<td>0.14</td>
<td>0.28</td>
<td></td>
<td>0.14</td>
</tr>
<tr>
<td>green translation</td>
<td>0.44</td>
<td>0.13</td>
<td>0.13</td>
<td></td>
<td>0.13</td>
</tr>
</tbody>
</table>

67
## MAP vs. Approximations

<table>
<thead>
<tr>
<th>Translation String</th>
<th>MAP</th>
<th>Viterbi</th>
<th>4-Best Crunching</th>
<th>Derivation</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>red translation</td>
<td>0.28</td>
<td>0.16</td>
<td>0.16</td>
<td></td>
<td>0.16</td>
</tr>
<tr>
<td>blue translation</td>
<td>0.28</td>
<td>0.14</td>
<td>0.28</td>
<td></td>
<td>0.14</td>
</tr>
<tr>
<td>green translation</td>
<td>0.44</td>
<td>0.13</td>
<td>0.13</td>
<td></td>
<td>0.13</td>
</tr>
</tbody>
</table>

- Exact MAP decoding under spurious ambiguity is **intractable** on HG
## MAP vs. Approximations

<table>
<thead>
<tr>
<th>translation string</th>
<th>MAP</th>
<th>Viterbi</th>
<th>4-best crunching</th>
<th>derivation</th>
<th>probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>red translation</td>
<td>0.28</td>
<td>0.16</td>
<td>0.16</td>
<td></td>
<td>0.16</td>
</tr>
<tr>
<td>blue translation</td>
<td>0.28</td>
<td>0.14</td>
<td>0.28</td>
<td></td>
<td>0.14</td>
</tr>
<tr>
<td>green translation</td>
<td>0.44</td>
<td>0.13</td>
<td>0.13</td>
<td></td>
<td>0.13</td>
</tr>
</tbody>
</table>

- Exact MAP decoding under spurious ambiguity is **intractable** on HG
- Viterbi and crunching are efficient, but ignore most derivations
## MAP vs. Approximations

<table>
<thead>
<tr>
<th>Translation String</th>
<th>MAP</th>
<th>Viterbi</th>
<th>4-Best Crunching</th>
<th>Derivation</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>red translation</td>
<td>0.28</td>
<td>0.16</td>
<td>0.16</td>
<td></td>
<td>0.16</td>
</tr>
<tr>
<td>blue translation</td>
<td>0.28</td>
<td>0.14</td>
<td>0.28</td>
<td></td>
<td>0.14</td>
</tr>
<tr>
<td>green translation</td>
<td>0.44</td>
<td>0.13</td>
<td>0.13</td>
<td></td>
<td>0.12</td>
</tr>
</tbody>
</table>

- Exact MAP decoding under spurious ambiguity is **intractable** on HG
- Viterbi and crunching are efficient, but ignore most derivations
- Our goal: develop an **approximation** that considers **all** the derivations but still allows **tractable** decoding
Variational Decoding
Variational Decoding

Decoding using Variational approximation

Decoding using a sentence-specific approximate distribution
Variational Decoding for MT: an Overview
Variational Decoding for MT: an Overview

Sentence-specific decoding
Variational Decoding for MT: an Overview

Sentence-specific decoding

Three steps:
Variational Decoding for MT: an Overview

Sentence-specific decoding

Three steps:

1. Generate a hypergraph for the foreign sentence
Variational Decoding for MT: an Overview

Sentence-specific decoding

Three steps:

1. Generate a hypergraph for the foreign sentence

Foreign sentence $x$
Variational Decoding for MT: an Overview

Sentence-specific decoding

Three steps:

1. Generate a hypergraph for the foreign sentence

Foreign sentence $x \rightarrow$ SMT
Variational Decoding for MT: an Overview

Sentence-specific decoding

Three steps:

1. Generate a hypergraph for the foreign sentence

MAP decoding under P is intractable

Foreign sentence $x$  \rightarrow  SMT  \rightarrow  

$\mathbf{p}(y, d \mid x)$

$\mathbf{p}(y \mid x)$

$X$  \rightarrow  $\langle dianzi, shang, the \ mat \rangle$

$X$  \rightarrow  $\langle mao, a \ cat \rangle$

$dianzi_0$  \quad  $shang_1$  \quad  $de_2$  \quad  $mao_3$
Generate a hypergraph
Generate a hypergraph
Generate a hypergraph
Generate a hypergraph
Generate a hypergraph

Estimate a model from the hypergraph

\[ q^*(y | x) \]
Generate a hypergraph

Estimate a model from the hypergraph

$q^*$ is an n-gram model over output strings.
Generate a hypergraph

Estimate a model from the hypergraph

$q^*(y | x) \approx \sum_{d \in D(x,y)} p(d | x)$

$q^*$ is an n-gram model over output strings.
Generate a hypergraph

Estimate a model from the hypergraph

$q^*$ is an n-gram model over output strings.

$q^*(y \mid x) \approx \sum_{d \in D(x,y)} p(d \mid x)$
Generate a hypergraph

Estimate a model from the hypergraph

 Decode using $q^*$ on the hypergraph

$q^*(y | x) \approx \sum_{d \in D(x,y)} p(d|x)$
Variational Inference
Variational Inference

• We want to do inference under $p$, but it is intractable
Variational Inference

- We want to do inference under $p$, but it is intractable

$$y^* = \arg \max_y p(y|x)$$
Variational Inference

• We want to do inference under $p$, but it is intractable

$$y^* = \arg \max_y p(y|x)$$

• Instead, we derive a simpler distribution $q^*$
Variational Inference

- We want to do inference under $p$, but it is intractable

$$y^* = \arg \max_y p(y|x)$$

- Instead, we derive a simpler distribution $q^*$

$$q^* = \arg \min_{q \in Q} \text{KL}(p||q)$$
Variational Inference

• We want to do inference under $p$, but it is intractable

$$y^* = \arg \max_y p(y|x)$$

• Instead, we derive a simpler distribution $q^*$

$$q^* = \arg \min_{q \in Q} \text{KL}(p||q)$$

• Then, we will use $q^*$ as a surrogate for $p$ in inference
Variational Inference

• We want to do inference under $p$, but it is intractable

$$y^* = \arg \max_y p(y|x)$$

• Instead, we derive a simpler distribution $q^*$

$$q^* = \arg \min_{q \in Q} KL(p||q)$$

• Then, we will use $q^*$ as a surrogate for $p$ in inference

$$y^* = \arg \max_y q^*(y \mid x)$$
Variational Inference

• We want to do inference under $p$, but it is intractable

$$y^* = \arg \max_y p(y|x)$$

• Instead, we derive a simpler distribution $q^*$

$$q^* = \arg \min_{q \in Q} \text{KL}(p||q)$$

• Then, we will use $q^*$ as a surrogate for $p$ in inference

$$y^* = \arg \max_y q^*(y \mid x)$$
Variational Inference

- We want to do inference under $p$, but it is intractable

$$y^* = \arg \max_y p(y|x)$$

- Instead, we derive a simpler distribution $q^*$

$$q^* = \arg \min_{q \in Q} KL(p || q)$$

- Then, we will use $q^*$ as a surrogate for $p$ in inference

$$y^* = \arg \max_y q^*(y \mid x)$$
Variational Inference

• We want to do inference under $p$, but it is intractable

$$y^* = \arg \max_y p(y | x)$$

• Instead, we derive a simpler distribution $q^*$

$$q^* = \arg \min_{q \in Q} \text{KL}(p || q)$$

• Then, we will use $q^*$ as a surrogate for $p$ in inference

$$y^* = \arg \max_y q^*(y | x)$$
Variational Inference

- We want to do inference under $p$, but it is intractable

$$y^* = \arg \max_y p(y|x)$$

- Instead, we derive a simpler distribution $q^*$

$$q^* = \arg \min_{q \in Q} \text{KL}(p || q)$$

- Then, we will use $q^*$ as a surrogate for $p$ in inference

$$y^* = \arg \max_y q^*(y | x)$$
Variational Inference

- We want to do inference under $p$, but it is intractable

  $$y^* = \arg \max_y p(y | x)$$

- Instead, we derive a simpler distribution $q^*$

  $$q^* = \arg \min_{q \in Q} KL(p || q)$$

- Then, we will use $q^*$ as a surrogate for $p$ in inference

  $$y^* = \arg \max_y q^*(y | x)$$
Variational Approximation

- \( q^* \): an approximation having minimum distance to \( p \)

\[
q^* = \arg \min_{q \in Q} KL(p || q)
\]

a family of distributions
Variational Approximation

- $q^*$: an approximation having minimum distance to $p$

$$q^* = \arg\min_{q \in Q} \text{KL}(p\|q)$$

$$= \arg\min_{q \in Q} \sum_{y \in \text{Trans}(x)} p \log \frac{p}{q}$$

a family of distributions
Variational Approximation

- $q^*$: an approximation having minimum distance to $p$

$$ q^* = \arg \min_{q \in Q} KL(p \| q) $$

$$ = \arg \min_{q \in Q} \sum_{y \in \text{Trans}(x)} p \log \frac{p}{q} $$

$$ = \arg \min_{q \in Q} \sum_{y \in \text{Trans}(x)} (p \log p - p \log q) $$

- a family of distributions
Variational Approximation

- $q^*$: an approximation having minimum distance to $p$

\[
q^* = \arg\min_{q \in Q} \text{KL}(p||q) = \arg\min_{q \in Q} \sum_{y \in \text{Trans}(x)} p \log \frac{p}{q}
\]

\[
= \arg\min_{q \in Q} \sum_{y \in \text{Trans}(x)} (p \log p - p \log q)
\]

a family of distributions  
constant
**Variational Approximation**

- $q^*$: an approximation having minimum distance to $p$

\[
q^* = \arg\min_{q \in Q} \text{KL}(p \| q)
\]

\[
= \arg\min_{q \in Q} \sum_{y \in \text{Trans}(x)} p \log \frac{p}{q}
\]

\[
= \arg\min_{q \in Q} \sum_{y \in \text{Trans}(x)} (p \log p - p \log q)
\]

\[
= \arg\max_{q \in Q} \sum_{y \in \text{Trans}(x)} p \log q
\]

A family of distributions

- Constant
Variational Approximation

- $q^*$: an approximation having minimum distance to $p$

$$q^* = \arg\min_{q \in Q} KL(p \| q)$$

$$= \arg\min_{q \in Q} \sum_{y \in \text{Trans}(x)} p \log \frac{p}{q}$$

$$= \arg\min_{q \in Q} \sum_{y \in \text{Trans}(x)} (p \log p - p \log q)$$

$$= \arg\max_{q \in Q} \sum_{y \in \text{Trans}(x)} p \log q$$

- Three questions

A family of distributions

Constant
Variational Approximation

• \( q^\ast \): an approximation having minimum distance to \( p \)

\[
q^\ast = \arg\min_{q \in Q} \text{KL}(p \| q)
\]

\[
= \arg\min_{q \in Q} \sum_{y \in \text{Trans}(x)} p \log \frac{p}{q}
\]

\[
= \arg\min_{q \in Q} \sum_{y \in \text{Trans}(x)} (p \log p - p \log q)
\]

\[
= \arg\max_{q \in Q} \sum_{y \in \text{Trans}(x)} p \log q
\]

• Three questions

• how to parameterize \( q \)?

• a family of distributions

• constant
Variational Approximation

• \( q^* \): an approximation having minimum distance to \( p \)

\[
q^* = \arg \min_{q \in Q} \text{KL}(p || q)
\]

\[
= \arg \min_{q \in Q} \sum_{y \in \text{Trans}(x)} p \log \frac{p}{q}
\]

\[
= \arg \min_{q \in Q} \sum_{y \in \text{Trans}(x)} (p \log p - p \log q)
\]

\[
= \arg \max_{q \in Q} \sum_{y \in \text{Trans}(x)} p \log q
\]

\[
\text{a family of distributions}
\]

• Three questions

• how to parameterize \( q \)?

• how to estimate \( q^* \)?
Variational Approximation

- \( q^* \): an approximation having minimum distance to \( p \)

\[
q^* = \arg \min_{q \in Q} \text{KL}(p \| q)
\]

\[
= \arg \min_{q \in Q} \sum_{y \in \text{Trans}(x)} p \log \frac{p}{q}
\]

\[
= \arg \min_{q \in Q} \sum_{y \in \text{Trans}(x)} (p \log p - p \log q)
\]

\[
= \arg \max_{q \in Q} \sum_{y \in \text{Trans}(x)} p \log q
\]

- Three questions
  - how to parameterize \( q \)?
  - how to estimate \( q^* \)?
  - how to use \( q^* \) for decoding?
**Variational Approximation**

- \( q^* \): an approximation having minimum distance to \( p \)

\[
q^* = \arg \min_{q \in Q} \text{KL}(p \| q)
\]

\[
= \arg \min_{q \in Q} \sum_{y \in \text{Trans}(x)} p \log \frac{p}{q}
\]

\[
= \sum_{y \in \text{Trans}(x)} (p \log p - p \log q)
\]

\[
= \arg \max_{q \in Q} \sum_{y \in \text{Trans}(x)} p \log q
\]

- Three questions
  - how to parameterize \( q \)?
  - how to estimate \( q^* \)?
  - how to use \( q^* \) for decoding?

*an n-gram model*
Variational Approximation

- $q^*$: an approximation having minimum distance to $p$

\[
q^* = \underset{q \in Q}{\arg \min} \, KL(p \parallel q)
\]

\[
= \underset{q \in Q}{\arg \min} \sum_{y \in \text{Trans}(x)} \frac{p \log p}{q}
\]

\[
= \underset{q \in Q}{\arg \min} \sum_{y \in \text{Trans}(x)} \left( p \log p - q \log q \right)
\]

\[
= \underset{q \in Q}{\arg \max} \sum_{y \in \text{Trans}(x)} q \log q
\]

- Three questions
  - how to parameterize $q$?
  - how to estimate $q^*$?
  - how to use $q^*$ for decoding?

- an n-gram model
- compute expected n-gram counts and normalize
Variational Approximation

- \( q^* \): an approximation having minimum distance to \( p \)

\[
q^* = \arg\min_{q \in Q} KL(p \| q)
\]

\[
= \arg\min_{q \in Q} \sum_{y \in \text{Trans}(x)} p \log \frac{p}{q}
\]

\[
= \arg\min_{q \in Q} \sum_{y \in \text{Trans}(x)} (p \log p - p \log q)
\]

\[
= \arg\max_{q \in Q} \sum_{y \in \text{Trans}(x)} p \log q
\]

- Three questions
  - how to parameterize \( q \)?
  - how to estimate \( q^* \)?
  - how to use \( q^* \) for decoding?

- an n-gram model
  - compute expected n-gram counts and normalize
  - score the hypergraph with the n-gram model
KL divergences under different variational models

\[ q^* = \arg \min_{q \in Q} KL(p \| q) = H(p, q) - H(p) \]
KL divergences under different variational models

\[ q^* = \arg \min_{q \in Q} \text{KL}(p \| q) = H(p, q) - H(p) \]

<table>
<thead>
<tr>
<th>Measure</th>
<th>( \overline{H}(p) )</th>
<th>( \overline{\text{KL}}(p | \cdot) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>bits/word</td>
<td>( q_1^* )</td>
<td>( q_2^* )</td>
</tr>
<tr>
<td>MT’04</td>
<td>1.36</td>
<td>0.97</td>
</tr>
<tr>
<td>MT’05</td>
<td>1.37</td>
<td>0.94</td>
</tr>
</tbody>
</table>
KL divergences under different variational models

\[ q^* = \arg \min_{q \in Q} \text{KL}(p \| q) = \text{H}(p, q) - \text{H}(p) \]

<table>
<thead>
<tr>
<th>Measure</th>
<th>( \text{H}(p) )</th>
<th>( q_1^* )</th>
<th>( q_2^* )</th>
<th>( q_3^* )</th>
<th>( q_4^* )</th>
</tr>
</thead>
<tbody>
<tr>
<td>MT’04</td>
<td>1.36</td>
<td>0.97</td>
<td>0.32</td>
<td>0.21</td>
<td>0.17</td>
</tr>
<tr>
<td>MT’05</td>
<td>1.37</td>
<td>0.94</td>
<td>0.32</td>
<td>0.21</td>
<td>0.17</td>
</tr>
</tbody>
</table>

- Larger \( n \) \( \Rightarrow \) better approximation \( q_n \) \( \Rightarrow \) smaller KL divergence from \( p \)

- The reduction of KL divergence happens mostly when switching from unigram to bigram
## BLEU Results on Chinese-English

### NIST MT 2004 Tasks

<table>
<thead>
<tr>
<th>Decoding scheme</th>
<th>BLEU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Viterbi</td>
<td>35.4</td>
</tr>
<tr>
<td>MBR (K=1000)</td>
<td>35.8</td>
</tr>
<tr>
<td>Crunching (N=10000)</td>
<td>35.7</td>
</tr>
<tr>
<td>Crunching+MBR (N=10000)</td>
<td>35.8</td>
</tr>
<tr>
<td>Variational ((1\text{to}4\text{gram}+wp+vt))</td>
<td>36.6</td>
</tr>
</tbody>
</table>

- variational decoding improves over Viterbi, MBR, and crunching

*(Kumar and Byrne, 2004)*

*(May and Knight, 2006)*
Variational Inference

• We want to do inference under $p$, but it is intractable

$$y^* = \arg \max_y p(y|x)$$

• Instead, we derive a simpler distribution $q^*$

$$q^* = \arg \min_{q \in Q} \text{KL}(p \| q)$$

• Then, we will use $q^*$ as a surrogate for $p$ in inference

$$y^* = \arg \max_y q^*(y \mid x)$$
Variational Inference

- We want to do inference under $p$, but it is intractable

\[ y^* = \arg \max_y p(y|x) \]

- Instead, we derive a simpler distribution $q^*$

\[ q^* = \arg \min_{q \in Q} \text{KL}(p||q) \]

- Then, we will use $q^*$ as a surrogate for $p$ in inference

\[ y^* = \arg \max_y q^*(y \mid x) \]
Variational Inference

• We want to do inference under $p$, but it is intractable

\[
y^* = \arg \max_y p(y|x)
\]

• Instead, we derive a simpler distribution $q^*$

\[
q^* = \arg \min_{q \in Q} KL(p||q)
\]

• Then, we will use $q^*$ as a surrogate for $p$ in inference

\[
y^* = \arg \max_y q^*(y | x)
\]
Outline

- Hypergraph as Hypothesis Space
- Unsupervised Discriminative Training
  - minimum imputed risk
  - contrastive language model estimation
- Variational Decoding

- First- and Second-order Expectation Semirings

**decoding**
(e.g., mbr)

**training**
(e.g., mert)

**atomic inference operations**
(e.g., finding one-best, k-best or expectation, inference can be exact or approximate)
A semiring framework to compute all of these

- **“decoding” quantities:**
  - Viterbi
  - K-best
  - Counting
  - ......

- **First-order quantities:**
  - expectation
  - entropy
  - Bayes risk
  - cross-entropy
  - KL divergence
  - feature expectations
  - first-order gradient of $Z$

- **Second-order quantities:**
  - expectation over product
  - interaction between features
  - Hessian matrix of $Z$
  - second-order gradient descent
  - gradient of expectation
  - gradient of entropy or Bayes risk
Compute Quantities on a Hypergraph: a Recipe

• Semiring-weighted inside algorithm
  • three steps:
Compute Quantities on a Hypergraph: a Recipe

- Semiring-weighted inside algorithm
- three steps:
  - choose a semiring
Compute Quantities on a Hypergraph: a Recipe

• Semiring-weighted inside algorithm

• three steps:
  ▶ choose a semiring

▶ specify a weight for each hyperedge
Compute Quantities on a Hypergraph: a Recipe

• Semiring-weighted inside algorithm
  • three steps:
    ▶ choose a semiring
    ▶ specify a weight for each hyperedge
    ▶ run the inside algorithm
Compute Quantities on a Hypergraph: a Recipe

- Semiring-weighted inside algorithm
- three steps:
  - choose a semiring \( \langle K, \oplus, \otimes \rangle \)
  - specify a weight for each hyperedge
  - run the inside algorithm
Compute Quantities on a Hypergraph: a Recipe

- Semiring-weighted inside algorithm
  - three steps:
    ▶ choose a semiring
    \[ \langle K, \oplus, \otimes \rangle \]
    - a set with **plus** and **times** operations
      - e.g., integer numbers with regular \(+\) and \(\times\)
    ▶ specify a weight for each hyperedge
    ▶ run the inside algorithm
Compute Quantities on a Hypergraph: a Recipe

• Semiring-weighted inside algorithm
  • three steps:
    ▶ choose a semiring
      \[ \langle K, \oplus, \otimes \rangle \]
      a set with **plus** and **times** operations
      e.g., integer numbers with regular + and ×
    ▶ specify a weight for each hyperedge
      each weight is a semiring member
    ▶ run the inside algorithm
Compute Quantities on a Hypergraph: a Recipe

- Semiring-weighted inside algorithm
- three steps:
  - choose a semiring
  - specify a weight for each hyperedge
  - run the inside algorithm

\[ \langle K, \oplus, \otimes \rangle \]

A set with **plus** and **times** operations

E.g., integer numbers with regular \(+\) and \(\times\)

- specify a weight for each hyperedge
  - each weight is a semiring member
- run the inside algorithm
  - complexity is \(O(\text{hypergraph size})\)
Semirings

- “Decoding” time semirings \(\text{(Goodman, 1999)}\)
  - counting, Viterbi, K-best, etc.
- “Training” time semirings
  - first-order expectation semirings \(\text{(Eisner, 2002)}\)
  - second-order expectation semirings \(\text{(new)}\)
- Applications of the Semirings \(\text{(new)}\)
  - entropy, risk, gradient of them, and many more
How many trees?
How many trees?

four 😊
How many trees? four 😊
compute it use a semiring?

How many trees?
Compute the Number of Derivation Trees

Three steps:
Compute the Number of Derivation Trees

Three steps:

- choose a semiring
Compute the Number of Derivation Trees

Three steps:

- choose a semiring
  - counting semiring:
    - ordinary integers with regular + and x
Compute the Number of Derivation Trees

Three steps:

- choose a semiring
  - counting semiring: ordinary integers with regular + and x
- specify a weight for each hyperedge
Compute the Number of Derivation Trees

Three steps:

- choose a semiring
  - counting semiring: ordinary integers with regular + and x
- specify a weight for each hyperedge
Compute the Number of Derivation Trees

Three steps:

- Choose a semiring
  - Counting semiring: ordinary integers with regular + and x
- Specify a weight for each hyperedge
- Run the inside algorithm
Bottom-up process in computing the number of trees
$k(v_1) = k(e_1)$

**Bottom-up** process in computing the number of trees
Bottom-up process in computing the number of trees

\[ k(v_1) = k(e_1) \]

\[ k(v_2) = k(e_2) \]
\( k(\mathbf{v}_1) = k(\mathbf{e}_1) \quad k(\mathbf{v}_2) = k(\mathbf{e}_2) \)
\( k(\mathbf{v}_3) = k(\mathbf{e}_3) \times k(\mathbf{v}_1) \times k(\mathbf{v}_2) \oplus k(\mathbf{e}_4) \times k(\mathbf{v}_1) \times k(\mathbf{v}_2) \)

**Bottom-up**
process in computing the number of trees
$k(v_1) = k(e_1)$

$k(v_2) = k(e_2)$

$k(v_3) = k(e_3) \otimes k(v_1) \otimes k(v_2) \oplus k(e_4) \otimes k(v_1) \otimes k(v_2)$

$k(v_4) = k(e_5) \otimes k(v_1) \otimes k(v_2) \oplus k(e_6) \otimes k(v_1) \otimes k(v_2)$

**Bottom-up**

process in computing the number of trees
Bottom-up process in computing the number of trees
Bottom-up process in computing the number of trees
expected translation length?

- **p=1/8**: dianzi\(_0\) shang\(_1\) de\(_2\) mao\(_3\)
  - X→dianzi shang, the mat
  - X→mao, a cat
  - X→(dianzi shang, the mat)
  - X→(mao, a cat)
  - X→(dianzi shang, the mat)
  - X→(mao, a cat)
- **p=2/8**: dianzi\(_0\) shang\(_1\) de\(_2\) mao\(_3\)
  - X→(dianzi shang, the mat)
  - X→(mao, a cat)
  - X→(dianzi shang, the mat)
  - X→(mao, a cat)
- **p=3/8**: dianzi\(_0\) shang\(_1\) de\(_2\) mao\(_3\)
  - X→(dianzi shang, the mat)
  - X→(mao, a cat)
  - X→(dianzi shang, the mat)
  - X→(mao, a cat)
  - X→(dianzi shang, the mat)
  - X→(mao, a cat)
  - X→(dianzi shang, the mat)
  - X→(mao, a cat)

**The Mat's a Cat**

- **p=2/8**: dianzi\(_0\) shang\(_1\) de\(_2\) mao\(_3\)
  - X→(dianzi shang, the mat)
  - X→(mao, a cat)
  - X→(dianzi shang, the mat)
  - X→(mao, a cat)
  - X→(dianzi shang, the mat)
  - X→(mao, a cat)
  - X→(dianzi shang, the mat)
  - X→(mao, a cat)
  - X→(dianzi shang, the mat)
  - X→(mao, a cat)

**A Cat on the Mat**

- **p=5/8**: dianzi\(_0\) shang\(_1\) de\(_2\) mao\(_3\)
  - X→(dianzi shang, the mat)
  - X→(mao, a cat)
  - X→(dianzi shang, the mat)
  - X→(mao, a cat)
  - X→(dianzi shang, the mat)
  - X→(mao, a cat)
  - X→(dianzi shang, the mat)
  - X→(mao, a cat)
  - X→(dianzi shang, the mat)
  - X→(mao, a cat)
The diagram illustrates various possible translations of a sentence in Chinese to English. The nodes represent words, with the edges showing the grammatical relationships and probabilities of different translations. The expected translation length is calculated as follows:

\[
\frac{2}{8} \times 4 + \frac{6}{8} \times 5 = 4.75
\]

The probabilities are:
- \(p=2/8\) for the mat a cat
- \(p=3/8\) for the mat's a cat
- \(p=1/8\) for a cat on the mat
- \(p=2/8\) for a cat of the mat
The expected translation length is calculated as follows:

\[ \frac{2}{8} \times 4 + \frac{6}{8} \times 5 = 4.75 \]

The variance is also calculated using the same approach.
expected translation length?

\[
2/8 \times 4 + 6/8 \times 5 = 4.75
\]

variance?

\[
2/8 \times (4-4.75)^2 + 6/8 \times (5-4.75)^2 \approx 0.19
\]
First-order:  

- each member is a 2-tuple:  \( \langle p, r \rangle \)

| \( \langle p_1, r_1 \rangle \otimes \langle p_2, r_2 \rangle \) | \( \langle p_1 p_2, p_1 r_2 + p_2 r_1 \rangle \) |
| \( \langle p_1, r_1 \rangle \oplus \langle p_2, r_2 \rangle \) | \( \langle p_1 + p_2, r_1 + r_2 \rangle \) |

Second-order:  

- each member is a 4-tuple:  \( \langle p, r, s, t \rangle \)

| \( \langle p_1, r_1, s_1, t_1 \rangle \otimes \langle p_2, r_2, s_2, t_2 \rangle \) | \( \langle p_1 p_2, p_1 r_2 + p_2 r_1, p_1 s_2 + p_2 s_1, p_1 t_2 + p_2 t_1 + r_1 s_2 + r_2 s_1 \rangle \) |
| \( \langle p_1, r_1, s_1, t_1 \rangle \oplus \langle p_2, r_2, s_2, t_2 \rangle \) | \( \langle p_1 + p_2, r_1 + r_2, s_1 + s_2, t_1 + t_2 \rangle \) |
\[k(v_1) = k(e_1) \quad k(v_2) = k(e_2)\]
\[k(v_3) = k(e_3) \otimes k(v_1) \otimes k(v_2) \oplus k(e_4) \otimes k(v_1) \otimes k(v_2)\]
\[k(v_4) = k(e_5) \otimes k(v_1) \otimes k(v_2) \oplus k(e_6) \otimes k(v_1) \otimes k(v_2)\]
\[k(v_5) = k(e_7) \otimes k(v_3) \oplus k(e_8) \otimes k(v_4)\]

**First-order:**
each semiring member is a 2-tuple
\[ k(v_1) = k(e_1) \]
\[ k(v_2) = k(e_2) \]
\[ k(v_3) = k(e_3) \otimes k(v_1) \otimes k(v_2) \oplus k(e_4) \otimes k(v_1) \otimes k(v_2) \]
\[ k(v_4) = k(e_5) \otimes k(v_1) \otimes k(v_2) \oplus k(e_6) \otimes k(v_1) \otimes k(v_2) \]
\[ k(v_5) = k(e_7) \otimes k(v_3) \oplus k(e_8) \otimes k(v_4) \]

Second-order: each semiring member is a 4-tuple
\[ k(v_1) = k(e_1) \quad \quad \quad k(v_2) = k(e_2) \]
\[ k(v_3) = k(e_3) \otimes k(v_1) \otimes k(v_2) \oplus k(e_4) \otimes k(v_1) \otimes k(v_2) \]
\[ k(v_4) = k(e_5) \otimes k(v_1) \otimes k(v_2) \oplus k(e_6) \otimes k(v_1) \otimes k(v_2) \]
\[ k(v_5) = k(e_7) \otimes k(v_3) \oplus k(e_8) \otimes k(v_4) \]

**Second-order:** each semiring member is a 4-tuple
Expectations on Hypergraphs
Expectations on Hypergraphs

• Expectation over a hypergraph
Expectations on Hypergraphs

- Expectation over a hypergraph

\[ \bar{r} \overset{\text{def}}{=} \mathbb{E}_p[r] = \sum_{d \in \text{HG}} p(d) r(d) \]

- \( r(d) \) is a function over a derivation \( d \)
e.g., the length of the translation yielded by \( d \)
Expectations on Hypergraphs

- Expectation over a hypergraph

\[ \bar{r} \overset{\text{def}}{=} \mathbb{E}_p[r] = \sum_{d \in \text{HG}} p(d) r(d) \]

- \( r(d) \) is a function over a derivation \( d \)
e.g., the length of the translation yielded by \( d \)

exponential size
Expectations on Hypergraphs

- Expectation over a hypergraph

\[
\bar{r} \overset{\text{def}}{=} \mathbb{E}_p[r] = \sum_{d \in \text{HG}} p(d) r(d)
\]

- \(r(d)\) is a function over a derivation \(d\)
  - e.g., the length of the translation yielded by \(d\)

- \(r(d)\) is additively decomposed

\[
r(d) \overset{\text{def}}{=} \sum_{e \in d} r_e
\]

- e.g., translation length is additively decomposed!
Second-order Expectations on Hypergraphs

• **Expectation of products** over a hypergraph

\[ \bar{t} \overset{\text{def}}{=} \mathbb{E}_p[r \cdot s] = \sum_{d \in \text{HG}} p(d) r(d) s(d) \]

• \( r \) and \( s \) are additively decomposed

\[ r(d) \overset{\text{def}}{=} \sum_{e \in d} r_e \]

\[ s(d) \overset{\text{def}}{=} \sum_{e \in d} s_e \]

\( r \) and \( s \) can be identical or different functions.
Compute expectation using expectation semiring:
Compute expectation using expectation semiring:

\[ k_e \overset{\text{def}}{=} \langle p_e, p_e r_e \rangle \]

\( p_e \): transition probability or log-linear score at edge \( e \)

\( r_e \)?
Compute expectation using expectation semiring:

\[ k_e \overset{\text{def}}{=} \langle p_e, p_e r_e \rangle \]

- \( p_e \): transition probability or log-linear score at edge \( e \)
- \( r_e \): 

\[ r_e \overset{\text{def}}{=} \log p_e \]

Entropy:
Compute expectation using expectation semiring:

\[ k_e \overset{\text{def}}{=} \langle p_e, p_e r_e \rangle \]

\( p_e \): transition probability or log-linear score at edge \( e \)

\( r_e \): transition probability or log-linear score at edge \( e \)

Entropy:

\[ r_e \overset{\text{def}}{=} \log p_e \]

Why?
Compute expectation using expectation semiring:

\[ k_e \overset{\text{def}}{=} \langle p_e, p_e r_e \rangle \]

\( p_e \): transition probability or log-linear score at edge \( e \)

\( r_e \)?

**Entropy:**

\[ r_e \overset{\text{def}}{=} \log p_e \]

**Why?** entropy is an **expectation**
Compute expectation using expectation semiring:

\[ k_e \overset{\text{def}}{=} \langle p_e, p_e r_e \rangle \]

\( p_e \): transition probability or log-linear score at edge e

\( r_e \)

Entropy:

\[ r_e \overset{\text{def}}{=} \log p_e \]

Why?

entropy is an **expectation**

\[ H(p) = \mathbb{E}_p [- \log p] = - \sum_{d \in HG} p(d) \log p(d) \]
Compute expectation using expectation semiring:

\[ k_e \overset{\text{def}}{=} \langle p_e, p_e r_e \rangle \]

\( p_e \): transition probability or log-linear score at edge \( e \)
\( r_e ? \)

**Entropy:**

\[ r_e \overset{\text{def}}{=} \log p_e \]

**Why?**

entropy is an **expectation**

\[
H(p) = \mathbb{E}_p[-\log p] = - \sum_{d \in HG} p(d) \log p(d)
\]

\( \log p(d) \) is additively decomposed!
Compute expectation using expectation semiring:

\[ k_e \overset{\text{def}}{=} \langle p_e, p_e r_e \rangle \]

\( p_e \): transition probability or log-linear score at edge \( e \)

\( r_e \) ?

**Entropy:**

\[ r_e \overset{\text{def}}{=} \log p_e \]

**Cross-entropy:**

\[ r_e \overset{\text{def}}{=} \log q_e \]

**Why?**
Compute expectation using expectation semiring:

\[ k_e \overset{\text{def}}{=} \langle p_e, p_e r_e \rangle \]

\( p_e \): transition probability or log-linear score at edge \( e \)

\( r_e \)?

Entropy:

\[ r_e \overset{\text{def}}{=} \log p_e \]

Cross-entropy:

\[ r_e \overset{\text{def}}{=} \log q_e \]

Why? cross-entropy is an expectation
Compute expectation using expectation semiring:

\[ k_e \overset{\text{def}}{=} \langle p_e, p_e r_e \rangle \]

\( p_e \): transition probability or log-linear score at edge \( e \)

\( r_e \) ?

**Entropy:**

\[ r_e \overset{\text{def}}{=} \log p_e \]

**Cross-entropy:**

\[ r_e \overset{\text{def}}{=} \log q_e \]

**Why?**

**cross-entropy is an expectation**

\[ H(p, q) = \mathbb{E}_p (- \log q) = - \sum_{d \in \text{HG}} p(d) \log q(d) \]
Compute expectation using expectation semiring:

\[ k_e \overset{\text{def}}{=} \langle p_e, p_e r_e \rangle \]

\( p_e \): transition probability or log-linear score at edge \( e \)
\( r_e \)?

**Entropy:**

\[ r_e \overset{\text{def}}{=} \log p_e \]

**Cross-entropy:**

\[ r_e \overset{\text{def}}{=} \log q_e \]

Why?

**cross-entropy is an expectation**

\[ H(p, q) = \mathbb{E}_p(-\log q) = -\sum_{d \in HG} p(d) \log q(d) \]

\( \log q(d) \) is additively decomposed!
Compute expectation using expectation semiring:

\[ k_e \overset{\text{def}}{=} \langle p_e, p_e r_e \rangle \]

\( p_e \) : transition probability or log-linear score at edge \( e \)

\( r_e ? \)

**Entropy:**

\( r_e \overset{\text{def}}{=} \log p_e \)

**Cross-entropy:**

\( r_e \overset{\text{def}}{=} \log q_e \)

**Bayes risk:**

\( r_e \overset{\text{def}}{=} \text{loss at edge } e \)

Why?
Compute expectation using expectation semiring:

\[ k_e \overset{\text{def}}{=} \langle p_e, p_e r_e \rangle \]

\( p_e \): transition probability or log-linear score at edge \( e \)
\( r_e \)?

**Entropy:**

\[ r_e \overset{\text{def}}{=} \log p_e \]

**Cross-entropy:**

\[ r_e \overset{\text{def}}{=} \log q_e \]

**Bayes risk:**

\[ r_e \overset{\text{def}}{=} \text{loss at edge } e \]

**Why?**

Bayes risk is an **expectation**
Compute expectation using expectation semiring:

\[ k_e \overset{\text{def}}{=} \langle p_e, p_e r_e \rangle \]

\( p_e \): transition probability or log-linear score at edge \( e \)

\( r_e \)?

**Entropy:**

\[ r_e \overset{\text{def}}{=} \log p_e \]

**Cross-entropy:**

\[ r_e \overset{\text{def}}{=} \log q_e \]

**Bayes risk:**

\[ r_e \overset{\text{def}}{=} \text{loss at edge } e \]

**Why?**

Bayes risk is an **expectation**

\[
\text{Risk} = \mathbb{E}_p(L) = - \sum_{d \in HG} p(d) \cdot L(Y(d))
\]
Compute expectation using expectation semiring:

\[ k_e \overset{\text{def}}{=} \langle p_e, p_e r_e \rangle \]

\( p_e \) : transition probability or log-linear score at edge \( e \)

\( r_e \) ?

**Entropy:**

\[ r_e \overset{\text{def}}{=} \log p_e \]

**Cross-entropy:**

\[ r_e \overset{\text{def}}{=} \log q_e \]

**Bayes risk:**

\[ r_e \overset{\text{def}}{=} \text{loss at edge } e \]

**Why?**

Bayes risk is an **expectation**

\[
\text{Risk} = \mathbb{E}_p(L) = - \sum_{d \in \text{HG}} p(d) \cdot L(Y(d))
\]

\( L(Y(d)) \) is additively decomposed!
Compute expectation using expectation semiring:

\[ k_e \overset{\text{def}}{=} \langle p_e, pe r_e \rangle \]

\( p_e \): transition probability or log-linear score at edge \( e \)

\( r_e \)?

**Entropy:**

\[ r_e \overset{\text{def}}{=} \log p_e \]

**Cross-entropy:**

\[ r_e \overset{\text{def}}{=} \log q_e \]

**Bayes risk:**

\[ r_e \overset{\text{def}}{=} \text{loss at edge } e \]

**Why?**

Bayes risk is an **expectation**

\[ \text{Risk} = \mathbb{E}_p(L) = - \sum_{d \in HG} p(d) \cdot L(Y(d)) \]

\( L(Y(d)) \) is additively decomposed! (Tromble et al. 2008)
### Applications of Expectation Semirings: a Summary

<table>
<thead>
<tr>
<th>Quantity</th>
<th>$k_e$</th>
<th>$k_{\text{root}}$</th>
<th>Final</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Expectation</strong></td>
<td>$\langle p_e, p_e r_e \rangle$</td>
<td>$\langle Z, \overline{r} \rangle$</td>
<td>$\overline{r} / Z$</td>
</tr>
<tr>
<td><strong>Entropy</strong></td>
<td>$r_e \text{ def } \log p_e$, so $k_e = \langle p_e, p_e \log p_e \rangle$</td>
<td>$\langle Z, \overline{r} \rangle$</td>
<td>$\log Z - \overline{r} / Z$</td>
</tr>
<tr>
<td>Cross-entropy</td>
<td>$r_e \text{ def } \log q_e$, so $k_e = \langle p_e, p_e \log q_e \rangle$</td>
<td>$\langle Z_q, \overline{r} \rangle$</td>
<td>$\log Z_q - \overline{r} / Z_p$</td>
</tr>
<tr>
<td>Bayes risk</td>
<td>$r_e \text{ def } L_e$, so $k_e = \langle p_e, p_e L_e \rangle$</td>
<td>$\langle Z, \overline{r} \rangle$</td>
<td>$\overline{r} / Z$</td>
</tr>
<tr>
<td><strong>First-order gradient</strong></td>
<td>$\langle p_e, \nabla p_e \rangle$</td>
<td>$\langle Z, \nabla Z \rangle$</td>
<td>$\nabla Z$</td>
</tr>
<tr>
<td><strong>Covariance matrix</strong></td>
<td>$\langle p_e, p_e r_e, p_e s_e, p_e r_e s_e \rangle$</td>
<td>$\langle Z, \overline{r}, \overline{s}, \overline{t} \rangle$</td>
<td>$\frac{\overline{t}}{Z} - \frac{\overline{r} \overline{s}^T}{Z^2}$</td>
</tr>
<tr>
<td><strong>Hessian matrix</strong></td>
<td>$\langle p_e, \nabla p_e, \nabla p_e, \nabla^2 p_e \rangle$</td>
<td>$\langle Z, \nabla Z, \nabla Z, \nabla^2 Z \rangle$</td>
<td>$\nabla^2 Z$</td>
</tr>
<tr>
<td><strong>Gradient of expectation</strong></td>
<td>$\langle p_e, p_e r_e, \nabla p_e, (\nabla p_e) r_e + p_e (\nabla r_e) \rangle$</td>
<td>$\langle Z, \overline{r}, \nabla Z, \nabla \overline{r} \rangle$</td>
<td>$\frac{Z \nabla \overline{r} - \overline{r} \nabla Z}{Z^2}$</td>
</tr>
<tr>
<td><strong>Gradient of entropy</strong></td>
<td>$\langle p_e, p_e \log p_e, \nabla p_e, (1 + \log p_e) \nabla p_e \rangle$</td>
<td>$\langle Z, \overline{r}, \nabla Z, \nabla \overline{r} \rangle$</td>
<td>$\frac{\nabla Z}{Z} - \frac{Z \nabla \overline{r} - \overline{r} \nabla Z}{Z^2}$</td>
</tr>
<tr>
<td><strong>Gradient of risk</strong></td>
<td>$\langle p_e, p_e L_e, \nabla p_e, L_e \nabla p_e \rangle$</td>
<td>$\langle Z, \overline{r}, \nabla Z, \nabla \overline{r} \rangle$</td>
<td>$\frac{Z \nabla \overline{r} - \overline{r} \nabla Z}{Z^2}$</td>
</tr>
</tbody>
</table>
A semiring framework to compute all of these

- **First-order quantities:**
  - expectation
  - entropy
  - Bayes risk
  - cross-entropy
  - KL divergence
  - feature expectations
  - first-order gradient of $Z$

- **Second-order quantities:**
  - Expectation over product
  - interaction between features
  - Hessian matrix of $Z$
  - second-order gradient descent
  - gradient of expectation
  - gradient of entropy or Bayes risk

- **“decoding” quantities:**
  - Viterbi
  - K-best
  - Counting
  - ......