Exact decoding for phrase-based SMT

Wilker Aziz[1], Marc Dymetman[2], Lucia Specia[1]

[1]University of Sheffield
[2]Xerox Research Centre Europe

October 27, 2014

Outline

Decoding

Viterbi decoding

$$
\begin{aligned}
\mathbf{d}^* &= \underset{\mathbf{d}\in\mathcal{D}(\mathbf{x})}{\operatorname{argmax}}\ f(\mathbf{d}) \\
&= \underset{\mathbf{d}\in\mathcal{D}(\mathbf{x})}{\operatorname{argmax}}\ \boldsymbol{\theta}^\top \mathbf{H}(\mathbf{d})
\end{aligned}
$$

Decoding

Viterbi decoding

$$\mathbf{d}^* = \operatorname*{argmax}_{\mathbf{d}\in\mathcal{D}(\mathbf{x})} f(\mathbf{d})$$
$$= \operatorname*{argmax}_{\mathbf{d}\in\mathcal{D}(\mathbf{x})} \boldsymbol{\theta}^\top \mathbf{H}(\mathbf{d})$$

space of translation derivations compatible with the input $\mathbf{x}$

Decoding

Viterbi decoding

$$\mathbf{d}^* = \operatorname*{argmax}_{\mathbf{d} \in \mathcal{D}(\mathbf{x})} f(\mathbf{d})$$
$$= \operatorname*{argmax}_{\mathbf{d} \in \mathcal{D}(\mathbf{x})} \boldsymbol{\theta}^\top \mathbf{H}(\mathbf{d})$$

we are looking for the best derivation
under a linear parameterisation ($\boldsymbol{\theta} \in \mathbb{R}^m$)

Decoding

Viterbi decoding

$$\mathbf{d}^* = \underset{\mathbf{d}\in\mathcal{D}(\mathbf{x})}{\operatorname{argmax}} \; f(\mathbf{d})$$
$$= \underset{\mathbf{d}\in\mathcal{D}(\mathbf{x})}{\operatorname{argmax}} \; \boldsymbol{\theta}_1^\top \mathbf{H}_1(\mathbf{d}) + \boldsymbol{\theta}_2^\top \mathbf{H}_2(\mathbf{d})$$

"local" features assess steps in a derivation independently

$$\mathbf{H}_1(\mathbf{d}) = \sum_{e\in\mathbf{d}} \mathbf{h}_1(e)$$

Decoding

Viterbi decoding

$$\mathbf{d}^* = \operatorname*{argmax}_{\mathbf{d} \in \mathcal{D}(\mathbf{x})} f(\mathbf{d})$$
$$= \operatorname*{argmax}_{\mathbf{d} \in \mathcal{D}(\mathbf{x})} \boldsymbol{\theta}_1^\top \mathbf{H}_1(\mathbf{d}) + \boldsymbol{\theta}_2^\top \mathbf{H}_2(\mathbf{d})$$

"local" features assess steps in a derivation independently

$$\mathbf{H}_1(\mathbf{d}) = \sum_{e \in \mathbf{d}} \mathbf{h}_1(e)$$

"nonlocal" features make weaker independence assumptions

e.g. $H_{\mathsf{LM}}(\mathbf{d}) = \log p_{\mathsf{LM}}(\text{yield}(\mathbf{d}))$

Complexity

$\langle \mathcal{D}(\mathbf{x}), f(\mathbf{d}) \rangle$ can be seen as the intersection between

Complexity

$\langle \mathcal{D}(\mathbf{x}), f(\mathbf{d}) \rangle$ can be seen as the intersection between

- a translation hypergraph $\mathcal{G}(\mathbf{x})$

locally parameterised

Complexity

$\langle \mathcal{D}(\mathbf{x}), f(\mathbf{d}) \rangle$ can be seen as the intersection between

- a translation hypergraph $\mathcal{G}(\mathbf{x})$

  locally parameterised

- and a target language model $\mathcal{A}$

  as a wFSA

Complexity

$\langle \mathcal{D}(\mathbf{x}), f(\mathbf{d}) \rangle$ can be seen as the intersection between

- a translation hypergraph $\mathcal{G}(\mathbf{x})$

locally parameterised

- and a target language model $\mathcal{A}$

as a wFSA

Phrase-based SMT with a distortion limit (d) and an n-gram LM

$$|\mathcal{G}(\mathbf{x})| \propto I^2 2^d$$

$$|\mathcal{A}| \propto |\Delta|^{n-1}$$

Complexity

$\langle \mathcal{D}(\mathbf{x}), f(\mathbf{d}) \rangle$ can be seen as the intersection between

- a translation hypergraph $\mathcal{G}(\mathbf{x})$

  locally parameterised

- and a target language model $\mathcal{A}$

  as a wFSA

Phrase-based SMT with a distortion limit (d) and an n-gram LM

$$|\mathcal{G}(\mathbf{x})| \propto I^2 2^d$$
$$|\mathcal{A}| \propto |\Delta|^{n-1}$$

Problem

The intersection is too large for standard dynamic programming

Contribution

Previous work on exact decoding

- compact models
- simpler parameterisation using 3-gram LMs

Contribution

Previous work on exact decoding

- compact models
- simpler parameterisation using 3-gram LMs

This work

- large models
- realistic 5-gram LMs

Intuition

Full intersection is wasteful

$$\mathcal{G}(\mathbf{x}) \cap \mathcal{A}$$

Intuition

Full intersection is wasteful

$$\mathcal{G}(\mathbf{x}) \cap \mathcal{A}$$

- complete n-gram LM

Intuition

Full intersection is wasteful

$$\mathcal{G}(\mathbf{x}) \cap \mathcal{A} = \mathcal{G}(\mathbf{x}) \cap \left(\bigcap_{i=1}^{M} \mathcal{A}^{(i)} \right)$$

- complete n-gram LM
- encodes one n-gram

Intuition

Full intersection is wasteful

$$\mathcal{G}(\mathbf{x}) \cap \mathcal{A} = \mathcal{G}(\mathbf{x}) \cap \left(\bigcap_{i=1}^{M} \mathcal{A}^{(i)} \right)$$

- complete n-gram LM
- encodes one n-gram

Assumption

not every n-gram participates in high-scoring derivations

Intuition

Full intersection is wasteful

$$\mathcal{G}(\mathbf{x}) \cap \mathcal{A} = \mathcal{G}(\mathbf{x}) \cap \left(\bigcap_{i=1}^{M} \mathcal{A}^{(i)} \right)$$

- complete n-gram LM
- encodes one n-gram

Assumption

not every n-gram participates in high-scoring derivations

Problem

which n-grams are really necessary?
and at which level of refinement?

Intuition

Full intersection is wasteful

$$\mathcal{G}(\mathbf{x}) \cap \mathcal{A} = \mathcal{G}(\mathbf{x}) \cap \left(\bigcap_{i=1}^{M} \mathcal{A}^{(i)} \right)$$

- complete n-gram LM
- encodes one n-gram

Assumption

not every n-gram participates in high-scoring derivations

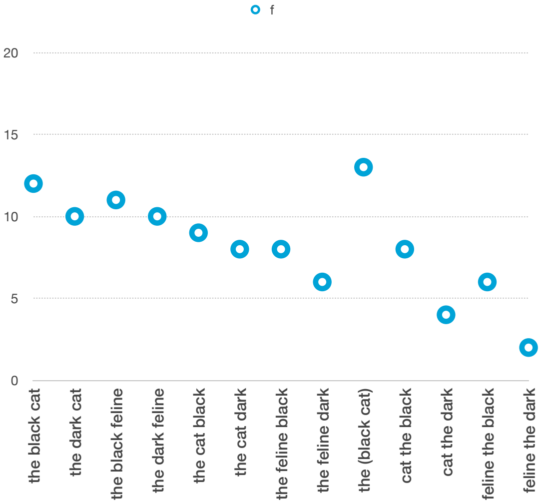
Problem

which n-grams are really necessary?
and at which level of refinement?

Strategy

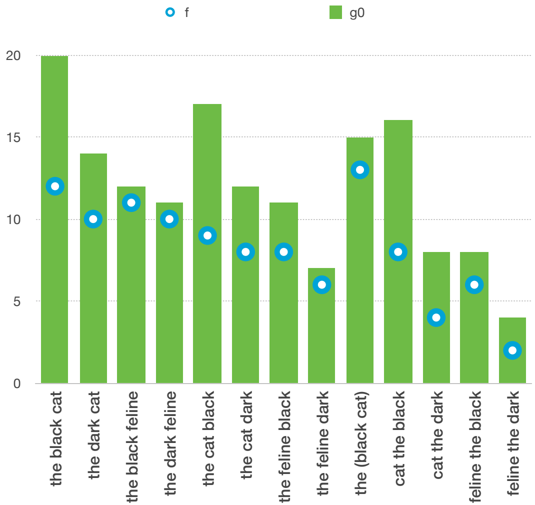
start with strong independence assumptions
revisit those assumptions as necessary

OS*



- upperbound the complex target $f(\mathbf{d})$

OS*

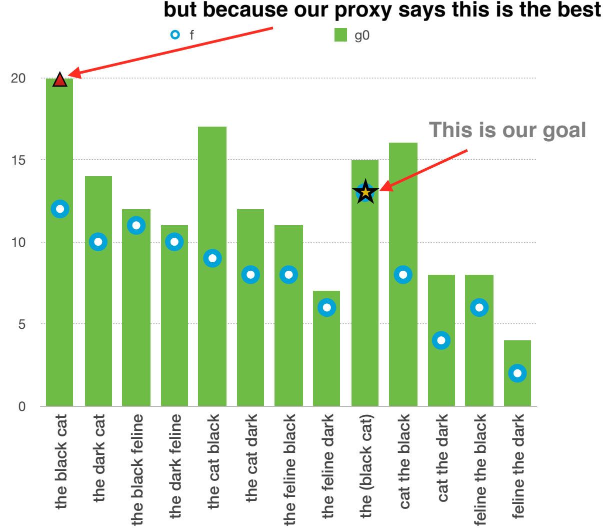


- upperbound the complex target $f(\mathbf{d})$ by a simpler proposal $g(\mathbf{d})$

OS*

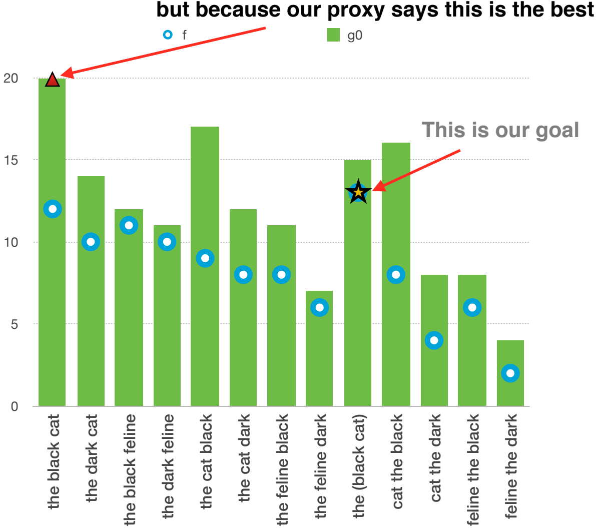
- we are interested in finding f's argmax
  however we cannot search through f

OS*



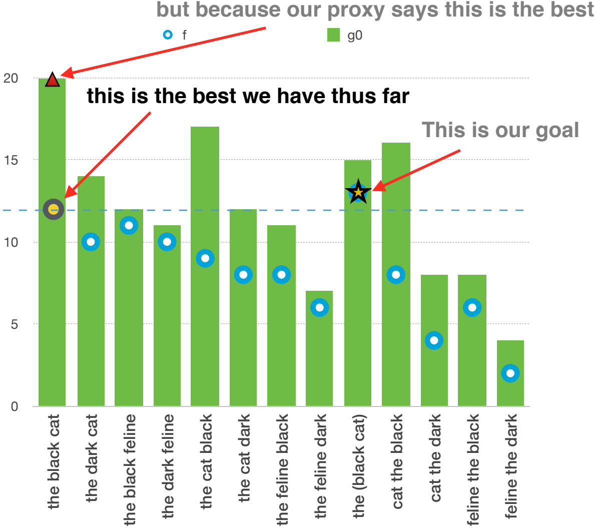
- we find $\mathbf{d}^* = \operatorname{argmax}_{\mathbf{d}}\, g(\mathbf{d})$

OS*



- we find $\mathbf{d}^* = \operatorname{argmax}_{\mathbf{d}}\, g(\mathbf{d})$ and assess $f(\mathbf{d}^*)$

OS*

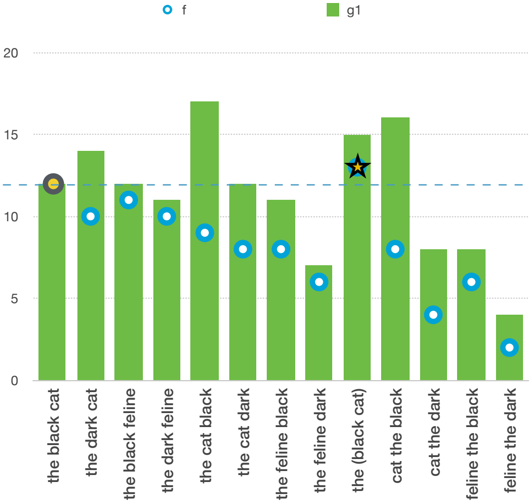


- we find $\mathbf{d}^* = \operatorname{argmax}_{\mathbf{d}} g(\mathbf{d})$ and assess $f(\mathbf{d}^*)$
- finding our best solution thus far (according to f)

OS*

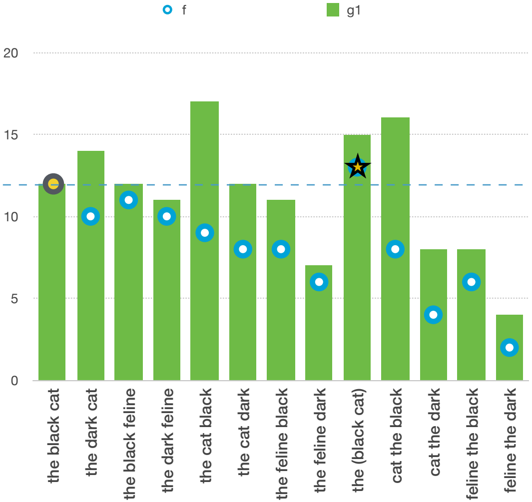
- we find $\mathbf{d}^* = \mathrm{argmax}_{\mathbf{d}}\, g(\mathbf{d})$ and assess $f(\mathbf{d}^*)$
- finding our best solution thus far (according to f)
- however we cannot guarantee exactness yet

OS*



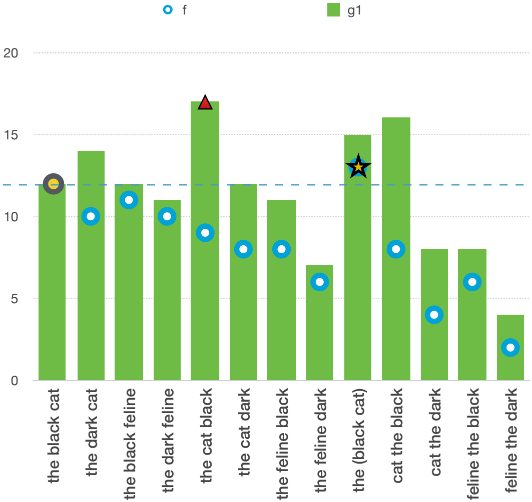
- so we refine g as to bring it closer to f
  e.g. by making $g(\mathbf{d}^*) = f(\mathbf{d}^*)$

OS*



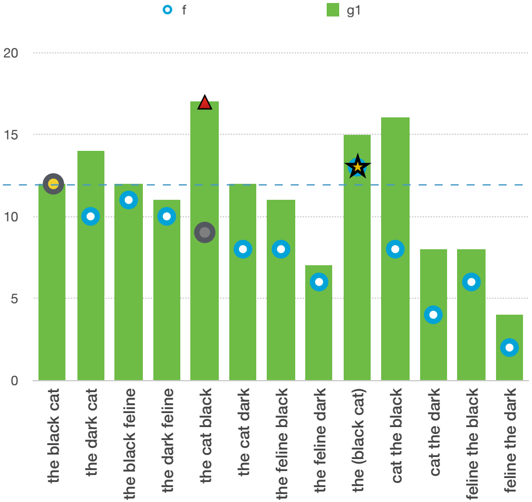
- so we refine g as to bring it closer to f
  e.g. by making $g(\mathbf{d}^*) = f(\mathbf{d}^*)$
- at the cost of some little complexity increase (extra nodes and edges)

OS*



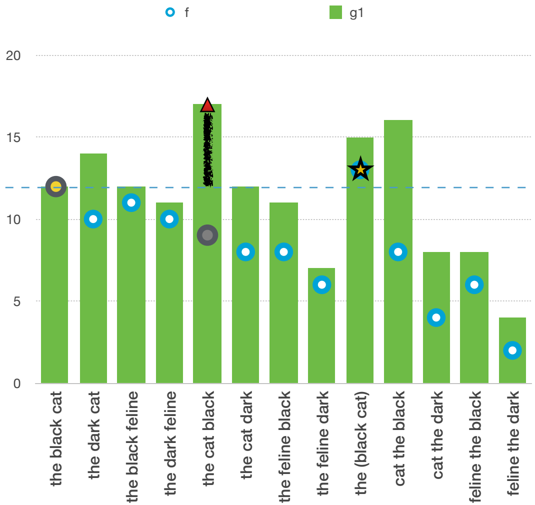
- as a consequence, g's argmax has changed
  we solve $\mathbf{d}^* = \mathrm{argmax}_{\mathbf{d}}\, g(\mathbf{d})$

OS*



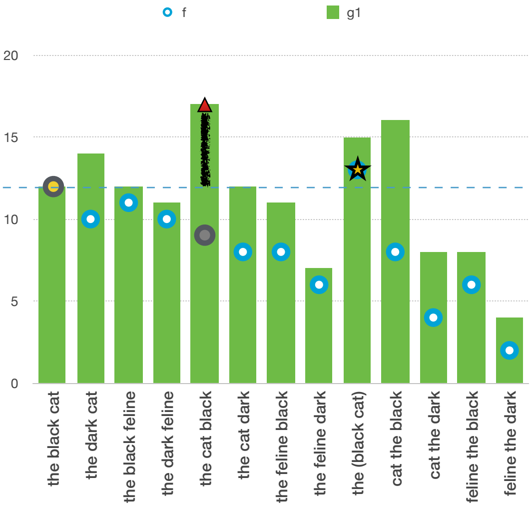
- as a consequence, g's argmax has changed
  we solve $\mathbf{d}^* = \mathrm{argmax}_{\mathbf{d}}\, g(\mathbf{d})$
- and assess $f(\mathbf{d}^*)$ again
  our best solution thus far might remain unchanged

OS*



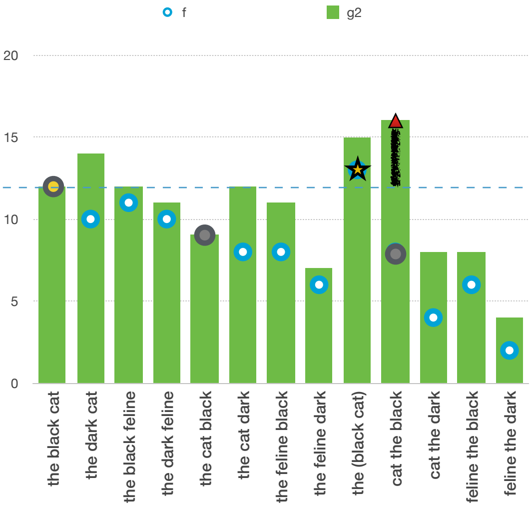
- we cannot yet guarantee exactness

OS*



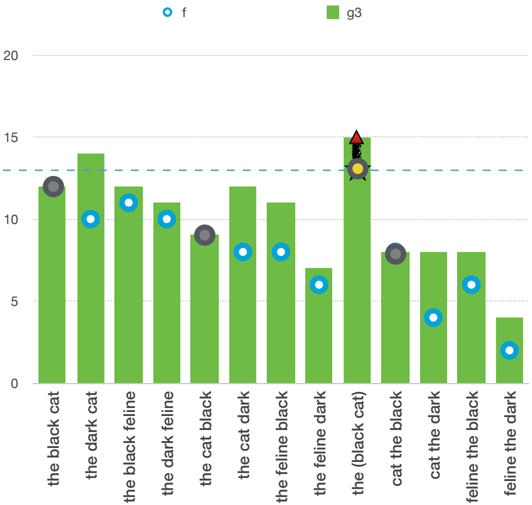
- we cannot yet guarantee exactness
- even though our maximum error is smaller

OS*



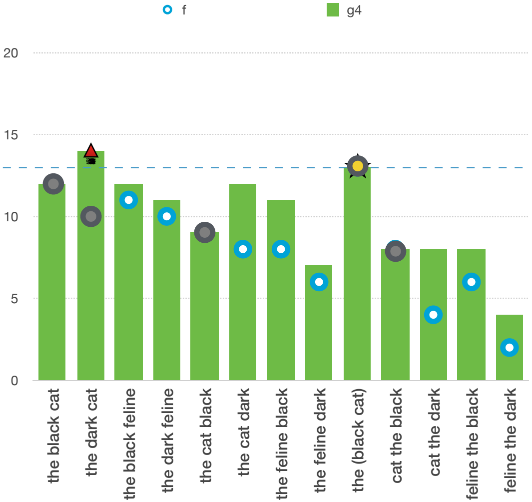
- but we can continue refining g

OS*



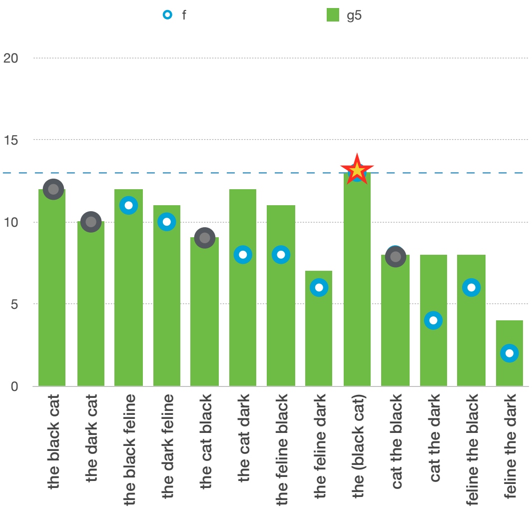
- but we can continue refining g
- for as long as g and f disagree on the maximum

OS*



- but we can continue refining g
- for as long as g and f disagree on the maximum

OS*



- but we can continue refining g
- for as long as g and f disagree on the maximum
- until we have a certificate of optimality

Decoding algorithm

function $\textsc{Optimise}(g, \epsilon)$

Decoding algorithm

1: **function** $\textsc{Optimise}(g, \epsilon)$
2: $\quad \mathbf{d}^* \leftarrow \mathrm{argmax}_{\mathbf{d}}\, g(\mathbf{d})$ ▷ proxy's argmax

Decoding algorithm

function $\textsc{Optimise}(g, \epsilon)$
$\quad \mathbf{d}^* \leftarrow \mathrm{argmax}_{\mathbf{d}}\, g(\mathbf{d})$ $\qquad \triangleright$ proxy's argmax
$\quad g^* \leftarrow g(\mathbf{d}^*)$
$\quad f^* \leftarrow f(\mathbf{d}^*)$ $\qquad \triangleright$ observe a point in f

Decoding algorithm

function $\textsc{Optimise}(g, \epsilon)$
  $\mathbf{d}^* \leftarrow \mathrm{argmax}_{\mathbf{d}}\, g(\mathbf{d})$ ▷ proxy's argmax
  $g^* \leftarrow g(\mathbf{d}^*)$
  $f^* \leftarrow f(\mathbf{d}^*)$ ▷ observe a point in f
  while $(q^* - f^* \geq \epsilon)$ **do** ▷ ϵ is the maximum error

Decoding algorithm

```
function OPTIMISE(g, ϵ)
    d* ← argmax_d g(d)                      ▷ proxy's argmax
    g* ← g(d*)
    f* ← f(d*)                              ▷ observe a point in f
    while (q* − f* ≥ ϵ) do                  ▷ ϵ is the maximum error
        A ← actions(g, d*)                  ▷ collect refinement actions
```

Decoding algorithm

```
function OPTIMISE(g, ε)
    d* ← argmax_d g(d)                ▷ proxy's argmax
    g* ← g(d*)
    f* ← f(d*)                        ▷ observe a point in f
    while (q* − f* ≥ ε) do            ▷ ε is the maximum error
        A ← actions(g, d*)            ▷ collect refinement actions
        g ← refine(g, A)              ▷ update proposal
```

Decoding algorithm

function $\textsc{Optimise}(g, \epsilon)$

  $\mathbf{d}^* \leftarrow \mathrm{argmax}_{\mathbf{d}}\, g(\mathbf{d})$ ▷ proxy's argmax

  $g^* \leftarrow g(\mathbf{d}^*)$

  $f^* \leftarrow f(\mathbf{d}^*)$ ▷ observe a point in f

  while $(q^* - f^* \geq \epsilon)$ **do** ▷ ϵ is the maximum error

    $A \leftarrow \mathrm{actions}(g, \mathbf{d}^*)$ ▷ collect refinement actions

    $g \leftarrow \mathrm{refine}(g, A)$ ▷ update proposal

    $\mathbf{d}^* \leftarrow \mathrm{argmax}_{\mathbf{d}}\, g(\mathbf{d})$ ▷ update argmax

Decoding algorithm

function $\textsc{Optimise}(g, \epsilon)$
  $\mathbf{d}^* \leftarrow \mathrm{argmax}_{\mathbf{d}}\, g(\mathbf{d})$ ▷ proxy's argmax
  $g^* \leftarrow g(\mathbf{d}^*)$
  $f^* \leftarrow f(\mathbf{d}^*)$ ▷ observe a point in f
  while $(q^* - f^* \geq \epsilon)$ **do** ▷ ϵ is the maximum error
    $A \leftarrow \mathrm{actions}(g, \mathbf{d}^*)$ ▷ collect refinement actions
    $g \leftarrow \mathrm{refine}(g, A)$ ▷ update proposal
    $\mathbf{d}^* \leftarrow \mathrm{argmax}_{\mathbf{d}}\, g(\mathbf{d})$ ▷ update argmax
    $g^* \leftarrow g(\mathbf{d}^*)$
    $f^* \leftarrow \max(f^*, f(\mathbf{d}^*))$ ▷ update "best so far"
  end while

Decoding algorithm

function $\textsc{Optimise}(g, \epsilon)$
  $\mathbf{d}^* \leftarrow \operatorname{argmax}_{\mathbf{d}} g(\mathbf{d})$ ▷ proxy's argmax
  $g^* \leftarrow g(\mathbf{d}^*)$
  $f^* \leftarrow f(\mathbf{d}^*)$ ▷ observe a point in f
  while $(q^* - f^* \geq \epsilon)$ **do** ▷ ϵ is the maximum error
    $A \leftarrow \text{actions}(g, \mathbf{d}^*)$ ▷ collect refinement actions
    $g \leftarrow \text{refine}(g, A)$ ▷ update proposal
    $\mathbf{d}^* \leftarrow \operatorname{argmax}_{\mathbf{d}} g(\mathbf{d})$ ▷ update argmax
    $g^* \leftarrow g(\mathbf{d}^*)$
    $f^* \leftarrow \max(f^*, f(\mathbf{d}^*))$ ▷ update "best so far"
  end while
  return g, $\mathbf{d}^*$
end function

Proposal

In g, the true LM p_{LM} is replaced by an upperbound q_{LM} with stronger independence assumptions

Proposal

In g, the true LM p_{LM} is replaced by an upperbound q_{LM}
with stronger independence assumptions

- Suppose $\alpha = y_I^J$ is a substring of $\mathbf{y} = y_1^M$
  e.g. $\alpha =$ black$_2$ cat$_3$ in $\mathbf{y} =$ BOS$_0$ the$_1$ black$_2$ cat$_3$ EOS$_4$

Proposal

In g, the true LM p_{LM} is replaced by an upperbound q_{LM}

with stronger independence assumptions

- Suppose $\alpha = y_I^J$ is a substring of $\mathbf{y} = y_1^M$

  e.g. $\alpha =$ black$_2$ cat$_3$ in $\mathbf{y} =$ BOS$_0$ the$_1$ black$_2$ cat$_3$ EOS$_4$

  - contribution of α to the true LM score of $\mathbf{y}$

$$p_{\text{LM}}(\alpha) \equiv \prod_{k=I}^{J} p(y_k | y_1^{k-1})$$

e.g. $p($black$_2|$BOS$_0$ the$_1)p($cat$_3|$BOS$_0$ the$_1$ black$_2)$

Proposal

In g, the true LM p_{LM} is replaced by an upperbound q_{LM} with stronger independence assumptions

- Suppose $\alpha = y_I^J$ is a substring of $\mathbf{y} = y_1^M$

  e.g. $\alpha =$ black$_2$ cat$_3$ in $\mathbf{y} =$ BOS$_0$ the$_1$ black$_2$ cat$_3$ EOS$_4$

  - contribution of α to the true LM score of $\mathbf{y}$

$$p_{\text{LM}}(\alpha) \equiv \prod_{k=I}^{J} p(y_k | y_1^{k-1})$$

e.g. $p(\text{black}_2 | \text{BOS}_0 \ \text{the}_1) p(\text{cat}_3 | \text{BOS}_0 \ \text{the}_1 \ \text{black}_2)$

  - upperbound to $p_{\text{LM}}(\alpha)$

$$q_{\text{LM}}(\alpha) \equiv q(y_I | \epsilon) \prod_{k=I+1}^{J} q(y_k | y_I^{k-1})$$

e.g. $q(\text{black}_2 | \epsilon) q(\text{cat}_3 | \text{black}_2)$

where $q(\mathtt{z} | \mathrm{P}) \equiv \max_{\mathtt{H} \in \Delta^*} p(\mathtt{z} | \mathtt{HP})$

Max-ARPA

An n-gram is scored in its most optimistic context $\mathtt{H}$

$$q(\mathtt{z}|\mathtt{P}) \equiv \max_{\mathtt{H}\in\Delta^*} p(\mathtt{z}|\mathtt{HP})$$

Max-ARPA

An n-gram is scored in its most optimistic context $\mathtt{H}$

$$q(\mathtt{z}|\mathtt{P}) \equiv \max_{\mathtt{H}\in\Delta^*} p(\mathtt{z}|\mathtt{HP})$$

Efficiently computed using a Max-ARPA table M

Max-ARPA

An n-gram is scored in its most optimistic context $\mathtt{H}$

$$q(\mathtt{z}|\mathtt{P}) \equiv \max_{\mathtt{H} \in \Delta^*} p(\mathtt{z}|\mathtt{HP})$$

Efficiently computed using a Max-ARPA table M

- start with an ARPA table

  n-gram $\mathtt{Pz}$ — conditional $\log p(\mathtt{z}|\mathtt{P})$ — backoff $b(\mathtt{Pz})$

Max-ARPA

An n-gram is scored in its most optimistic context $\mathtt{H}$

$$q(\mathtt{z}|\mathtt{P}) \equiv \max_{\mathtt{H}\in\Delta^*} p(\mathtt{z}|\mathtt{HP})$$

Efficiently computed using a Max-ARPA table M

- start with an ARPA table
  n-gram $\mathtt{Pz}$ — conditional $\log p(\mathtt{z}|\mathtt{P})$ — backoff $b(\mathtt{Pz})$
- compute an upperbound view of the last 2 columns
  n-gram $\mathtt{Pz}$ — max-conditional $\log q(\mathtt{z}|\mathtt{P})$ — max-backoff $m(\mathtt{Pz})$

Max-ARPA

An n-gram is scored in its most optimistic context $\mathtt{H}$

$$q(\mathtt{z}|\mathtt{P}) \equiv \max_{\mathtt{H}\in\Delta^*} p(\mathtt{z}|\mathtt{HP})$$

Efficiently computed using a Max-ARPA table M

- start with an ARPA table
  n-gram $\mathtt{Pz}$ — conditional $\log p(\mathtt{z}|\mathtt{P})$ — backoff $b(\mathtt{Pz})$
- compute an upperbound view of the last 2 columns
  n-gram $\mathtt{Pz}$ — max-conditional $\log q(\mathtt{z}|\mathtt{P})$ — max-backoff $m(\mathtt{Pz})$

Then for an arbitrary n-gram $\mathtt{Pz}$,

$$q(\mathtt{z}|\mathtt{P}) = \begin{cases} p(\mathtt{z}|\mathtt{P}) & \mathtt{Pz} \notin M \text{ and } \mathtt{P} \notin M \\ p(\mathtt{z}|\mathtt{P}) \times m(\mathtt{P}) & \mathtt{Pz} \notin M \text{ and } \mathtt{P} \in M \\ q(\mathtt{z}|\mathtt{P}) & \mathtt{Pz} \in M \end{cases}$$

Proposal hypergraph

The proposal $g(\mathbf{d})$ can be efficiently represented by a hypergraph

Proposal hypergraph

The proposal $g(\mathbf{d})$ can be efficiently represented by a hypergraph

Remark!

Nodes **do not** store LM state

Proposal hypergraph

The proposal $g(\mathbf{d})$ can be efficiently represented by a hypergraph

Remark!

Nodes **do not** store LM state

$$|\langle \mathcal{D}(\mathbf{x}), f(\mathbf{d}) \rangle| = |\mathcal{G}(\mathbf{d}) \cap \mathcal{A}| \propto I^2 2^d |\Delta|^{n-1}$$

Proposal hypergraph

The proposal $g(\mathbf{d})$ can be efficiently represented by a hypergraph

Remark!
Nodes **do not** store LM state

$$|\,\langle \mathcal{D}(\mathbf{x}), g(\mathbf{d}) \rangle\,| = |\mathcal{G}(\mathbf{d}) \cap \overset{\text{upperbound}}{\cancel{\mathcal{A}}}| \propto I^2 2^d \cancel{|\Delta|^{n-1}}$$

Refinement

Goal: break independence assumptions

1. making larger n-grams
2. bringing g closer to f

Refinement

Goal: break independence assumptions

1. making larger n-grams
2. bringing g closer to f

Examples

Refinement

Goal: break independence assumptions

1. making larger n-grams
2. bringing g closer to f

Examples

$$g'(\mathbf{d}) = \begin{cases} g(\mathbf{d}) & \text{if } \mathbf{d} \neq \operatorname{argmax} g(\mathbf{d}) \\ f(\mathbf{d}) & \text{otherwise.} \end{cases}$$

Refinement

Goal: break independence assumptions

1. making larger n-grams
2. bringing g closer to f

Examples

$$g'(\mathbf{d}) = \begin{cases} g(\mathbf{d}) & \text{if } \mathbf{d} \neq \operatorname{argmax} g(\mathbf{d}) \\ f(\mathbf{d}) & \text{otherwise.} \end{cases}$$

Too local: one derivation at a time

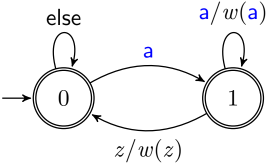
Refinement

Goal: break independence assumptions

1. making larger n-grams
2. bringing g closer to f

Examples

$$g'(\mathbf{d}) = g(\mathbf{d}) \frac{q(\mathtt{z}|\mathtt{hP})}{q(\mathtt{z}|\mathtt{P})}^{k}$$



where k counts occurrences of `hPz` in $\mathrm{yield}(\mathbf{d})$

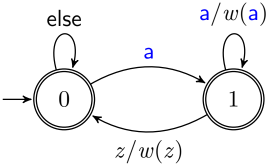
Refinement

Goal: break independence assumptions

1. making larger n-grams
2. bringing g closer to f

Examples

$$g'(\mathbf{d}) = g(\mathbf{d}) \frac{q(\mathtt{z}|\mathtt{hP})}{q(\mathtt{z}|\mathtt{P})}^{k}$$



where k counts occurrences of $\mathtt{hPz}$ in $\mathrm{yield}(\mathbf{d})$
Too global: refines derivations which already score poorly

LM state refinement[1]

Break independence assumptions (making larger n-grams)

but not in every derivation

[1] [Li and Khudanpur, 2008, Heafield et al., 2013]

LM state refinement[1]

Break independence assumptions (making larger n-grams)

but not in every derivation

Example:

- suppose the argmax is $(X_1(X_2(X_3\text{the})\text{black})\text{cat})$

[1] [Li and Khudanpur, 2008, Heafield et al., 2013]

LM state refinement[1]

Break independence assumptions (making larger n-grams)

but not in every derivation

Example:

- suppose the $\arg\max$ is $(X_1(X_2(X_3$the$)$black$)$cat$)$
- and node X_3 currently stores an empty LM states

[1] [Li and Khudanpur, 2008, Heafield et al., 2013]

LM state refinement[1]

Break independence assumptions (making larger n-grams)

but not in every derivation

Example:

- suppose the $\arg\max$ is $(X_1(X_2(X_3\text{the})\text{black})\text{cat})$
- and node X_3 currently stores an empty LM states
- this motivates a refined node $X_{3'}$ whose LM state is

$$\text{the} \cdot \textsc{LMState}(X_3)$$

[1] [Li and Khudanpur, 2008, Heafield et al., 2013]

LM state refinement[1]

Break independence assumptions (making larger n-grams)

but not in every derivation

Example:

- suppose the $\arg\max$ is $(X_1(X_2(X_3\text{the})\text{black})\text{cat})$
- and node X_3 currently stores an empty LM states
- this motivates a refined node $X_{3'}$ whose LM state is

$$\text{the} \cdot \text{LMS{\scriptsize TATE}}(X_3)$$

- incoming edges to X_3 are split

[1][Li and Khudanpur, 2008, Heafield et al., 2013]

LM state refinement[1]

Break independence assumptions (making larger n-grams)

but not in every derivation

Example:

- suppose the argmax is $(X_1(X_2(X_3\text{the})\text{black})\text{cat})$
- and node X_3 currently stores an empty LM states
- this motivates a refined node $X_{3'}$ whose LM state is

$$\text{the} \cdot \mathrm{LMS{\scriptsize TATE}}(X_3)$$

- incoming edges to X_3 are split
- outgoing edges from $X_{3'}$ are reweighted copies of those leaving X_3

[1] [Li and Khudanpur, 2008, Heafield et al., 2013]

LM state refinement[1]

Break independence assumptions (making larger n-grams)

but not in every derivation

Example:

- suppose the $\arg\max$ is $(X_1(X_2(X_3\text{the})\text{black})\text{cat})$
- and node X_3 currently stores an empty LM states
- this motivates a refined node $X_{3'}$ whose LM state is

$$\text{the} \cdot \text{LMSTATE}(X_3)$$

- incoming edges to X_3 are split
- outgoing edges from $X_{3'}$ are reweighted copies of those leaving X_3

This is connected to an intersection local to X_3

$*(X_3 * \text{the})z*$ with weight update $\frac{q(z|\text{the})}{q(z)}$

[1] [Li and Khudanpur, 2008, Heafield et al., 2013]

Refinement actions

Goal: lower g's maximum as much as possible

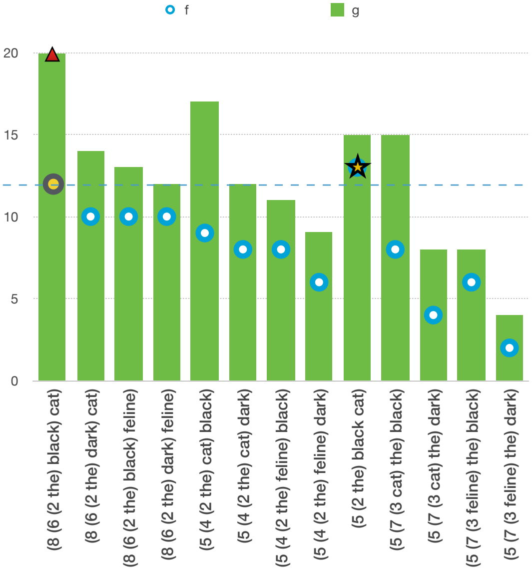
Refinement actions

Goal: lower g's maximum as much as possible

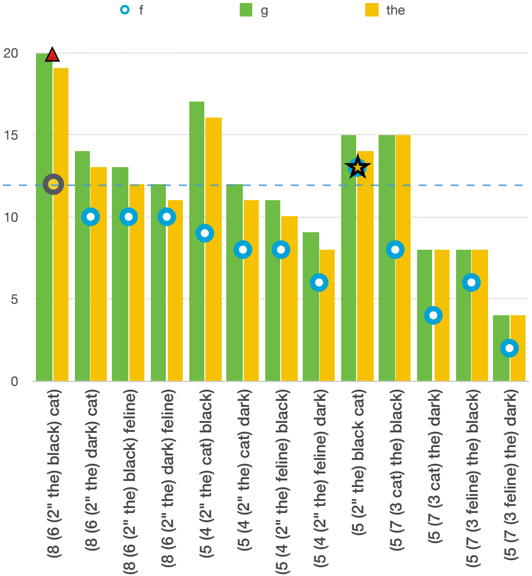
Heuristic

Refine **all nodes** participating in the **current** argmax

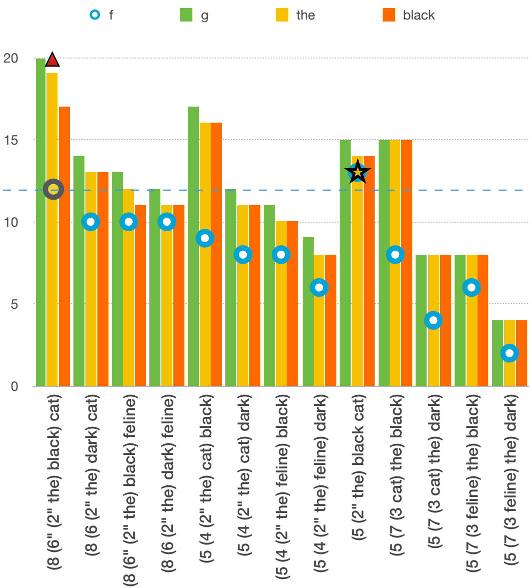
- by extending a node's LM (right) state by exactly one word from $\mathrm{yield}(\mathrm{argmax}\, g(\mathbf{d}))$



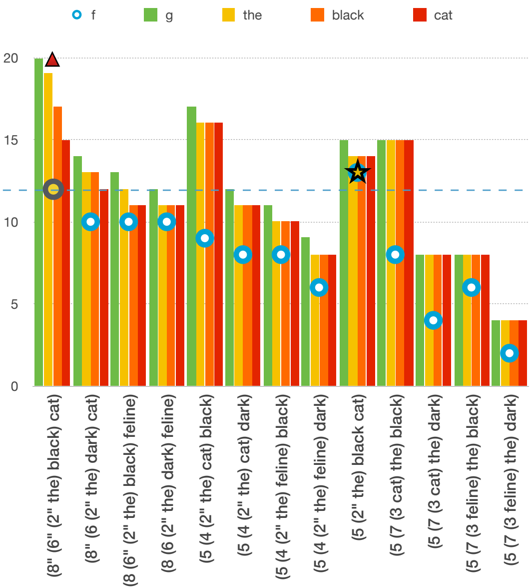
- argmax is $(X_8(X_6(X_2\text{the})\text{black})\text{cat})$



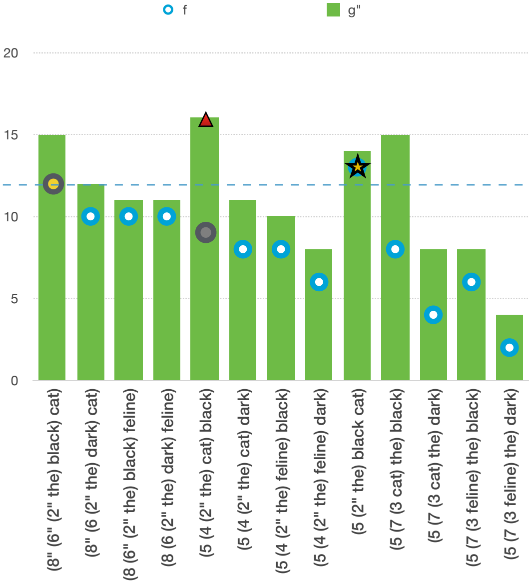
- the argmax is $(X_8(X_6(X_2\text{the})\text{black})\text{cat})$
- refine spans continuing from X_2 by conditioning on the



- the argmax is $(X_8(X_6(X_2\text{the})\text{black})\text{cat})$
- refine spans continuing from X_6 by conditioning on black



- the argmax is $(X_8(X_6(X_2\text{the})\text{black})\text{cat})$
- refine spans continuing from X_8 by conditioning on cat



- obtaining a new hypergraph $\langle \mathcal{D}(\mathbf{x}), g'(\mathbf{d}) \rangle$

Experiments

German-English (WMT14 data)

- phrase extraction: 2.2M sentences
- maximum phrase length 5
- maximum translation options 40
- unpruned LMs: 25M sentences
- dev set: $\textsc{newstest}2010$ (LM interpolation and tuning)
- batch-mira tuning (cube pruning beam 5000)
- test set: $\textsc{newstest}2012$ (3,003 sentences)
- distortion limit $d = 4$

Exact decoding

n	build (s)	total (s)	N	$\|V\|$	$\|E\|$
3	1.5	21	190	2.5	159
4	1.5	50	350	4	288
5	1.5	106	555	6.1	450

- time to build initial proposal
- decoding time
- number of iterations
- size of the hypergraph (in thousands of nodes and edges)

Cube pruning

Search errors by beam size (k)

k	n-gram LM		
	3	4	5
10	2168	2347	2377
10^2	613	999	1126
10^3	29	102	167
10^4	0	4	7

Translation quality with BLEU

Cube pruning and exact decoding with OS^*

k	3-gram LM	4-gram LM	5-gram LM
10	20.47	20.71	20.69
10^2	21.14	21.73	21.76
10^3	21.27	21.89	21.91
10^4	21.29	21.92	21.93
OS^*	21.29	21.92	21.93

Conclusions

Exact decoding

Conclusions

Exact decoding

- manageable time

Conclusions

Exact decoding

- manageable time
- fraction of the search space

Conclusions

Exact decoding

- manageable time
- fraction of the search space

Search error curves for beam search and cube pruning

Conclusions

Exact decoding

- manageable time
- fraction of the search space

Search error curves for beam search and cube pruning

- large phrase tables
- large 5-gram LMs

Conclusions

Exact decoding

- manageable time
- fraction of the search space

Search error curves for beam search and cube pruning

- large phrase tables
- large 5-gram LMs

Exactness at the cost of worst-case complexity, however

Conclusions

Exact decoding

- manageable time
- fraction of the search space

Search error curves for beam search and cube pruning

- large phrase tables
- large 5-gram LMs

Exactness at the cost of worst-case complexity, however

- we demonstrate empirically that the algorithm is practicable

Recent developments

1. exact k-best
2. exact sampling

Future work

Optimisation

- early stop the search
- error safe pruning

Future work

Optimisation

- early stop the search
- error safe pruning

Speedups

- be more selective with refinements
- LR to deal with powerset constraints (allowing for higher d)
- more grouping (partial edges)

Future work

Optimisation

- early stop the search
- error safe pruning

Speedups

- be more selective with refinements
- LR to deal with powerset constraints (allowing for higher d)
- more grouping (partial edges)

Hiero models

Thanks!

Questions?

Variable-order LM

n	Nodes at level m					LM states at level m			
	0	1	2	3	4	1	2	3	4
3	0.4	1.2	0.5	-	-	113	263	-	-
4	0.4	1.6	1.4	0.3	-	132	544	212	-
5	0.4	2.1	2.4	0.7	0.1	142	790	479	103

Table : Average number of nodes (in thousands) whose LM state encode an m-gram, and average number of unique LM states of order m in the final hypergraph for different n-gram LMs ($d = 4$ everywhere).

References I

Kenneth Heafield, Philipp Koehn, and Alon Lavie. Grouping language model boundary words to speed k-best extraction from hypergraphs. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 958–968, Atlanta, Georgia, USA, June 2013.

Zhifei Li and Sanjeev Khudanpur. A scalable decoder for parsing-based machine translation with equivalent language model state maintenance. In *Proceedings of the Second Workshop on Syntax and Structure in Statistical Translation*, SSST '08, pages 10–18, Stroudsburg, PA, USA, 2008. Association for Computational Linguistics.