Lecture 17: Issues in NLP and Possible Architectures for NLP
Lecture Plan

1. Solving language: High-level needs
2. Efficient tree-recursive models: SPINN and SNLI
3. Research highlight: Learning to compose for QA
4. Interlude: pointer/copying models
5. Sub-word and character-based models

Reminders/comments:

Best of luck in finishing your Ass 4 / Final Project!
Make sure you have solid baselines, well-trained models
Get help! Richard has infinite project office hours tonight!
Azure: We have credits, let us know if you have needs!
People are out to “solve” language

“The role of FAIR is to advance the science and the technology of AI and do experiments that demonstrate that technology for new applications like computer vision, dialogue systems, virtual assistants, speech recognition, natural language understanding, translation, things like that.”

Yann LeCun

http://www.businessinsider.com/interview-yann-lecun-facebook-ai-deepmind-2016-10
What has been lost from old NLP work?

An earlier era of work had lofty goals, but modest realities.

Today, we have much better realities, but often content ourselves with running LSTMs rather than reaching for the stars.
Peter Norvig’s thesis – 30th anniversary

A Unified Theory of Inference for Text Understanding

By

Peter Norvig
B.S. (Brown University) 1978

DISSERTATION

Submitted in partial satisfaction of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in

COMPUTER SCIENCE

in the

GRADUATE DIVISION

OF THE

UNIVERSITY OF CALIFORNIA, BERKELEY

Robert Wilensky
Lofti Zadeh
Chuck Fillmore

Approved: 
Chairman 
Date

11/25/86

11/25/86
In a poor fishing village built on an island not far from the coast of China, a young boy named Chang Lee lived with his widowed mother. Every day, little Chang bravely set off with his net, hoping to catch a few fish from the sea, which they could sell and have a little money to buy bread.

(a) There is a sea, which surrounds the island, is used by the villagers for fishing, and forms part of the coast of China

(b) Chang intends to trap fish in his net, which is a fishing net

(c) The word which refers to the fish

(d) The word they refers to Chang and his mother
The unified theory of inference

“As we have just seen, a suitable knowledge base is a prerequisite for making proper inferences” (p. 4). It’s built to enable inferences

System had 6 general forms of inference; 2 pairs, so 4 basic types:

1. Elaboration: Filling a slot to connect two entities
   • John got piggybank for REASON have money for REASON buy present

2. Reference Resolution: Hey – it’s coreference!!!

3. View Application: *The Red Sox killed the Yankees*
   • KILLED is not animal; KILLING is viewed as a DEFEAT-CONVINCINGLY

4. Concretization: Infer more specific
   • TRAVELLING in an AUTOMOBILE is an instance of DRIVING
Basic NLP: Progress has been made!

“Arens and Wilensky’s PHRAN program was used where possible [to convert input sentences to KODIAK knowledge representations]. For some input, PHRAN was not up to the task, so a representation was constructed by hand instead.” (p. 4)
Building elaborations a la Norvig (1986)
What do we still need?

BiLSTMs with attention seem to be taking over the field and improving our ability to do everything

Neural methods are leading to a renaissance for all language generation tasks (i.e., MT, dialog, QA, summarization, ...)

There’s a real scientific question of where and whether we need explicit, localist language and knowledge representations and inferential mechanisms
What do we still need?

However:

We still have very primitive methods for building and accessing memories or knowledge

Current models have almost nothing for developing and executing goals and plans

We still have quite inadequate abilities for understanding and using inter-sentential relationships

We still can’t, at a large scale, do elaborations from a situation using common sense knowledge
2. Political Ideology Detection Using Recursive Neural Networks

[Iyyer, Enns, Boyd-Graber & Resnik 2014]
Political Ideology Detection Using Tree Recursive Neural Networks

But taxpayers do know already that TARP was designed in a way that allowed the same corporations who were saved by huge amounts of taxpayer money to continue to show the same arrogant traits that should have destroyed their companies.
Tree recursive NNs (TreeRNNs)

- Theoretically appealing
- Empirically competitive
- But
- Prohibitively slow
- Often used with an external parser
- Don’t exploit complementary linear structure of language
A recurrent NN allows efficient batched computation on GPUs
TreeRNN: Input-specific structure undermines batched computation.
The Shift-reduce Parser-Interpreter NN (SPINN) [Bowman, Gauthier et al. 2016]

Base model is equivalent to a TreeRNN, but ...

supports batched computation: $25 \times$ speedups

Plus:

Effective new hybrid that combines linear and tree-structured context

Can stand alone without a parser
Beginning observation: binary trees = transition sequences
The Shift-reduce Parser-Interpreter NN (SPINN)
The Shift-reduce Parser-Interpreter NN (SPINNN)

The model includes a sequence LSTM RNN

- This acts as a simple parser by predicting SHIFT or REDUCE
- It also gives left sequence context as input to composition
Implementing the stack

- Naïve implementation: simulates stacks in a batch with a fixed-size multidimensional array at each timestep
  - Backpropagation requires that each intermediate stack be maintained in memory
  - $\Rightarrow$ Large amount of data copying and movement required
- Efficient implementation
  - Have only one stack array for each example
  - At each timestep, augment with the current head of the stack
  - Keep list of backpointers for REDUCE operations
- Similar to zipper data structures employed elsewhere
# A thinner stack

![Tree diagram](image)

<table>
<thead>
<tr>
<th>Array</th>
<th>Backpointers</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Spot</td>
</tr>
<tr>
<td>2</td>
<td>sat</td>
</tr>
<tr>
<td>3</td>
<td>down</td>
</tr>
<tr>
<td>4</td>
<td>(sat down)</td>
</tr>
<tr>
<td>5</td>
<td>(Spot (sat down))</td>
</tr>
</tbody>
</table>
A man rides a bike on a snow covered road.
A man is outside.  ENTAILMENT

2 female babies eating chips.
Two female babies are enjoying chips.
NEUTRAL

A man in an apron shopping at a market.
A man in an apron is preparing dinner.
CONTRADICTION
NLI with Tree-RNN sentence rep’ns
[Bowman, Angeli, Potts & Manning, EMNLP 2015]

Approach: We would like to work out the meaning of each sentence separately – a pure compositional model

Then we compare them with NN & classify for inference
Using SPINN for natural language inference

The cat sat down

The cat is angry

\( p(\text{entail} \mid h, p) \)

\( p(\text{contradict} \mid h, p) \)

\( p(\text{neutral} \mid h, p) \)
## SNLI Results

<table>
<thead>
<tr>
<th>Model</th>
<th>% Accuracy (Test set)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feature-based classifier</td>
<td>78.2</td>
</tr>
<tr>
<td>Previous SOTA sentence encoder</td>
<td>82.1</td>
</tr>
<tr>
<td>[Mou et al. 2016]</td>
<td></td>
</tr>
<tr>
<td>LSTM RNN sequence model</td>
<td>80.6</td>
</tr>
<tr>
<td>Tree LSTM</td>
<td>80.9</td>
</tr>
<tr>
<td><strong>SPINN</strong></td>
<td><strong>83.2</strong></td>
</tr>
<tr>
<td>SOTA (sentence pair alignment model)</td>
<td>86.8</td>
</tr>
<tr>
<td>[Parikh et al. 2016]</td>
<td></td>
</tr>
</tbody>
</table>
Successes for SPINN over LSTM

Examples with negation

• P: The rhythmic gymnast completes her floor exercise at the competition.
• H: The gymnast cannot finish her exercise.

Long examples (> 20 words)

• P: A man wearing glasses and a ragged costume is playing a Jaguar electric guitar and singing with the accompaniment of a drummer.
• H: A man with glasses and a disheveled outfit is playing a guitar and singing along with a drummer.
Learning to Compose Neural Networks for Question Answering

Authors: Jacob Andreas, Marcus Rohrbach, Trevor Darrell, Dan Klein

Research Highlight Presented by Zhedi Liu
High Level Overview

A compositional, attentional model for answering questions about a variety of world representations, including images and structured knowledge bases.
Two components, Trained Jointly

- Query: What cities are in Georgia?
- A collection of neural “modules” that can be freely composed
Two components, Trained Jointly

- Query: What cities are in Georgia?
- A network layout predictor that assembles modules into complete deep networks tailored to each question
Model: Built around Two Distributions

- **A Layout Model:** \( p(z|x; \theta_\ell) \)
  - chooses a layout for a sentence

- **An Execution Model:** \( p_z(y|w; \theta_e) \)
  - applies the network specified a particular layout to a world representation

---

1. \( w \) a world representation
2. \( x \) a question
3. \( y \) an answer
4. \( z \) a network layout
5. \( \theta \) a collection of model parameters
Step 1: Represent the input sentence as a dependency tree.

Query: What cities are in Georgia?
Step 2: Associate fragments of the dependency parse with appropriate modules

Query: What cities are in Georgia?
Layout Model

Step 3: Assemble fragments into full layouts

Query: What cities are in Georgia?
Layout Scoring Model

- Produce an LSTM representation of the question, a feature-based representation of the query, and pass both representations through a multilayer perceptron.

- The update to the layout-scoring model at each timestep is simply the gradient of the log-probability of the chosen layout, scaled by the accuracy of that layout’s predictions.
Query: What cities are in Georgia?
Module: lookup

Lookup

\[ [\text{lookup}[i]] = e_{f(i)} \]  \hspace{1cm} (2)

where \( e_i \) is the basis vector that is 1 in the \( i \)th position and 0 elsewhere.
Module: relate

Relate (Attention $\rightarrow$ Attention)

Relate directs focus from one region of the input to another. It behaves much like the \texttt{find} module, but also conditions its behavior on the current region of attention $h$. Let $\tilde{w}(h) = \sum_k h_k w^k$, where $h_k$ is the $k^{th}$ element of $h$. Then,

$$\left[\text{relate}[i](h)\right] = \text{softmax}(a \odot \sigma(Bv^i \oplus CW \oplus D\tilde{w}(h) \oplus e))$$

(4)
Module: find

Find

\[
\text{find}[i] \text{ computes a distribution over indices by concatenating the parameter argument with each position of the input feature map, and passing the concatenated vector through a MLP:}
\]

\[
\left[ \text{find}[i] \right] = \text{softmax}(a \odot \sigma(Bv^i \oplus CW \oplus d))
\]

(3)
Module: and

And \text{(Attention}^{*} \rightarrow \text{Attention}) \text{ and performs an operation analogous to set intersection for attentions. The analogy to probabilistic logic suggests multiplying probabilities:}

\[
\llbracket \text{and}(h^1, h^2, \ldots) \rrbracket = h^1 \odot h^2 \odot \ldots
\]  
(5)
Train an Execution Model

- Maximize $\sum_{(w, y, z)} \log p_z(y|w; \theta_e)$

1. $w$ a world representation
2. $x$ a question
3. $y$ an answer
4. $z$ a network layout
5. $\theta$ a collection of model parameters
State-of-the-art Performance: VQA

What is in the sheep’s ear?
(describe[what] (and find[sheep] find[ear]))
tag

What color is she wearing?
(describe[color] find[wear])
white

What is the man dragging?
(describe[what] find[man])
boat (board)
## State-of-the-art Performance: VQA

<table>
<thead>
<tr>
<th></th>
<th>test-dev</th>
<th></th>
<th></th>
<th>test-std</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Yes/No</td>
<td>Number</td>
<td>Other</td>
<td>All</td>
</tr>
<tr>
<td>Zhou (2015)</td>
<td>76.6</td>
<td>35.0</td>
<td>42.6</td>
<td>55.7</td>
</tr>
<tr>
<td>Noh (2015)</td>
<td>80.7</td>
<td>37.2</td>
<td>41.7</td>
<td>57.2</td>
</tr>
<tr>
<td>Yang (2015)</td>
<td>79.3</td>
<td>36.6</td>
<td>46.1</td>
<td>58.7</td>
</tr>
<tr>
<td>NMN</td>
<td>81.2</td>
<td>38.0</td>
<td>44.0</td>
<td>58.6</td>
</tr>
<tr>
<td>D-NMN</td>
<td>81.1</td>
<td>38.6</td>
<td>45.5</td>
<td>59.4</td>
</tr>
</tbody>
</table>
State-of-the-art Performance: GeoQA

<table>
<thead>
<tr>
<th>Question</th>
<th>Truth</th>
<th>Accuracy</th>
<th>GeoQA</th>
<th>GeoQA+Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>Is Key Largo an island?</td>
<td>yes: correct</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(exists (and lookup[key-largo] find[island]))</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>What national parks are in Florida?</td>
<td>everglades: correct</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(and find[park] (relate[in] lookup[florida]))</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>What are some beaches in Florida?</td>
<td>yes (daytona-beach):</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(exists (and lookup[beach] (relate[in] lookup[florida])))</td>
<td>wrong parse</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>What beach city is there in Florida?</td>
<td>[none] (daytona-beach):</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(and lookup[beach] lookup[city] (relate[in] lookup[florida]))</td>
<td>wrong module behavior</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Model</th>
<th>GeoQA</th>
<th>GeoQA+Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSP-F</td>
<td>48</td>
<td>–</td>
</tr>
<tr>
<td>LSP-W</td>
<td>51</td>
<td>–</td>
</tr>
<tr>
<td>NMN</td>
<td>51.7</td>
<td>35.7</td>
</tr>
<tr>
<td>D-NMN</td>
<td>54.3</td>
<td>42.9</td>
</tr>
</tbody>
</table>
4. Brief Interlude: Models with a pointer/copying

Recall the Pointer Sentinel Mixture Models (Merity et al. 2017) that Richard mentioned a few weeks ago.
Copying/pointer networks

- Gulcehre, Ahn, Nallapati, Zhou, Bengio (2016) Pointing the Unknown Words
Copying/pointer networks

- **MT:**
  
  Table 5: Europarl Dataset (EN-FR)
  
<table>
<thead>
<tr>
<th></th>
<th>BLEU-4</th>
</tr>
</thead>
<tbody>
<tr>
<td>NMT</td>
<td>20.19</td>
</tr>
<tr>
<td>NMT + PS</td>
<td>23.76</td>
</tr>
</tbody>
</table>

- Effective in tasks like summarization as well
- Caution from Google NMT paper: ‘In principle can train a “copy model” but this approach is both unreliable at scale – the attention mechanism is unstable when the network is deep – and copying may not always be the best strategy for rare words – sometimes transliteration is more appropriate’
5. Below the word: Writing systems

Most deep learning NLP work begins with language in its written form – it’s the easily processed, found data.

But human language writing systems aren’t one thing!

- Phonemic (maybe digraphs)
- Fossilized phonemic
- Syllabic/moraic
- Ideographic (syllabic)
- Combination of the above

jiyawu ngabulu
thorough failure
去年同期船二号坠毁
インド洋の島
Writing systems vary in how they represent words – or don’t

- No word segmentation
  - 美国关岛国际机场及其办公室均接获
- Words segmented
  - Clitics?
    - Separated
    - Joined
  - Compounds?
    - Separated
    - Joined

Je vous ai apporté des bonbons

فَقَلَ + نَا + هَا = فَقُلْنَاها

so+said+we+it

life insurance company employee

Lebensversicherungsgesellschaftsangestellter
Models below the word level

• Need to handle **large, open vocabulary**
  • Rich morphology: nejneobhospodařovávatelnějšímu ("to the worst farmable one")
• Informal spelling: **gooooooood morning !!!!!**
• Transliteration: **Christopher ➞ Kryštof**
Morphology

- Traditionally, have morpheme as smallest **semantic** unit
  - \([\text{un \ [[\text{fortun(e) } ]_{\text{ROOT}} \text{ ate}]}_{\text{STEM}}]_{\text{STEM}} \text{ ly}]_{\text{WORD}}\)

- Deep learning: Very little studied, though one attempt with recursive neural networks (Luong, Socher, & Manning 2013):

  Possible way of dealing with a larger vocabulary – most unseen words are new morphological forms (or numbers)
Morphology

- Alternative is to work with character n-grams
  - Wickelphones (Rumelhart & McClelland 1986)
  - Microsoft’s DSSM (Huang, He, Gao, Deng, Acero, & Hect 2013)
- Related to use of convolutional layers
- Can give many of the benefits of morphemes more easily??
Character-Level Models

Word embeddings can be composed from character embeddings

- Generates embeddings for unknown words
- Similar spellings share similar embeddings
- Solves OOV problem (ideally)

Has proven to work very successfully!

- Somewhat surprisingly – traditionally phonemes/letters weren’t a semantic unit
Character-based LSTM

Bi-LSTM builds word representations

(Unfortunately)

Character-based LSTM

(Unfortunately) the bank was closed

Recurrent Language Model

Bi-LSTM builds word representations

Used as LM and for POS tagging

Learning Character-level Representations for Part-of-Speech Tagging
Dos Santos and Zadrozny (2014)

- Convolution over characters to generate word embeddings
- Fixed window of word embeddings used for PoS tagging
Character-Aware Neural Language Models
(Kim, Jernite, Sontag, and Rush 2015)

- Character-based word embedding
- Utilizes convolution, highway network, and LSTM
A Joint Model for Word Embedding and Word Morphology
(Cao and Rei 2016)

- Same objective as w2v, but using characters
- Bi-directional LSTM to create embedding
- Model attempts to capture morphology
- Model can infer roots of words
Sub-word NMT: two trends

- **Same seq2seq architecture:**
  - Use smaller units.
  - [Sennrich, Haddow, Birch, ACL’16a], [Chung, Cho, Bengio, ACL’16].

- **Hybrid architectures:**
  - RNN for *words* + something else for *characters*.
  - [Costa-Jussà & Fonollosa, ACL’16], [Luong & Manning, ACL’16].
Byte Pair Encoding

• BPE is a compression algorithm:
  • Most frequent byte pair $\mapsto$ a new byte.

  Replace bytes with character ngrams

Byte Pair Encoding

- A word segmentation algorithm:
  - Start with a vocabulary of characters
  - Most frequent ngram pair $\mapsto$ a new ngram
Byte Pair Encoding

- A word segmentation algorithm:
  - Start with a vocabulary of characters
  - Most frequent ngram pair $\mapsto$ a new ngram

<table>
<thead>
<tr>
<th>Dictionary</th>
</tr>
</thead>
<tbody>
<tr>
<td>5  l  o  w</td>
</tr>
<tr>
<td>2  l  o  w  e  r</td>
</tr>
<tr>
<td>6  n  e  w  e  s  t</td>
</tr>
<tr>
<td>3  w  i  d  e  s  t</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Vocabulary</th>
</tr>
</thead>
<tbody>
<tr>
<td>l, o, w, e, r, n, w, s, t, i, d</td>
</tr>
</tbody>
</table>

Start with all characters in vocab

(Example from Sennrich)
Byte Pair Encoding

• A word segmentation algorithm:
  • Start with a vocabulary of characters
  • Most frequent ngram pair → a new ngram

Dictionary

| 5  | l o w |
| 2  | l o w e r |
| 6  | n e w e s t |
| 3  | w i d e s t |

Vocabulary

l, o, w, e, r, n, w, s, t, i, d, es

Add a pair (e, s) with freq 9

(Example from Sennrich)
Byte Pair Encoding

• A word segmentation algorithm:
  • Start with a vocabulary of characters
  • Most frequent ngram pair \( \mapsto \) a new ngram

**Dictionary**

<table>
<thead>
<tr>
<th>5</th>
<th>low</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>lower</td>
</tr>
<tr>
<td>6</td>
<td>new est</td>
</tr>
<tr>
<td>3</td>
<td>widest</td>
</tr>
</tbody>
</table>

**Vocabulary**

l, o, w, e, r, n, w, s, t, i, d, es, est

Add a pair (es, t) with freq 9

*(Example from Sennrich)*
Byte Pair Encoding

- A **word segmentation** algorithm:
  - Start with a vocabulary of **characters**
  - Most frequent **ngram pair** $\rightarrow$ a new **ngram**

**Dictionary**

<table>
<thead>
<tr>
<th>Rank</th>
<th>Token</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>lo w</td>
</tr>
<tr>
<td>2</td>
<td>lo w e r</td>
</tr>
<tr>
<td>6</td>
<td>new est</td>
</tr>
<tr>
<td>3</td>
<td>wid est</td>
</tr>
</tbody>
</table>

**Vocabulary**

l, o, w, e, r, n, w, s, t, i, d, es, est, lo

**Add a pair** (l, o) **with freq 7**

*Example from Sennrich*
Byte Pair Encoding

• A word segmentation algorithm:
  • Start with a vocabulary of characters
  • Most frequent ngram pair $\mapsto$ a new ngram

• Automatically decide vocab for NMT
  • No longer strongly “word” based in conventional way

Several top rankings in WMT 2016!

https://github.com/rsennrich/nematus
Wordpiece model

- GNMT uses a variant of this, the wordpiece model, which is generally similar but uses a greedy approximation to maximizing language model log likelihood to choose the pieces
Hybrid NMT

- A best-of-both-worlds architecture:
  - Translate mostly at the word level
  - Only go to the character level when needed.

- More than 2 BLEU improvement over a copy mechanism.

Hybrid NMT

Word-level (4 layers)

End-to-end training 8-stacking LSTM layers.
2-stage Decoding

- **Word-level beam search**
2-stage Decoding

- **Word-level** beam search
- **Char-level** beam search for `<unk>`.
## English-Czech Results

- Train on WMT’15 data (12M sentence pairs)
  - newstest2015

<table>
<thead>
<tr>
<th>Systems</th>
<th>BLEU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Winning WMT’15 (Bojar &amp; Tamchyna, 2015)</td>
<td>18.8</td>
</tr>
<tr>
<td><strong>Word-level</strong> NMT (Jean et al., 2015)</td>
<td>18.3</td>
</tr>
</tbody>
</table>

- 30x data
- 3 systems
- Large vocab
- + copy mechanism
English-Czech Results

- Train on WMT’15 data (12M sentence pairs)
  - newstest2015

<table>
<thead>
<tr>
<th>Systems</th>
<th>BLEU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Winning WMT’15 (Bojar &amp; Tamchyna, 2015)</td>
<td>18.8</td>
</tr>
<tr>
<td><strong>Word-level</strong> NMT (Jean et al., 2015)</td>
<td>18.3</td>
</tr>
<tr>
<td><strong>Hybrid</strong> NMT (Luong &amp; Manning, 2016)*</td>
<td><strong>20.7</strong></td>
</tr>
</tbody>
</table>

30x data
3 systems
Large vocab + copy mechanism
New SOTA!
Her 11-year-old daughter, Shani Bart, said it felt a little bit weird.

Její jedenáctiletá dcera Shani Bartová prozradila, že je to trochu zvláštní.

Její <unk> dcera <unk> <unk> řekla, že je to trochu divné.

Její 11-year-old dcera Shani, řekla, že je to trochu divné.

Její jedenáctiletá dcera, Graham Bart, řekla, že cítí trochu divný.

• Word-based: identity copy fails.
<table>
<thead>
<tr>
<th>source</th>
<th>Her 11-year-old daughter, Shani Bart, said it felt a little bit weird</th>
</tr>
</thead>
<tbody>
<tr>
<td>human</td>
<td>Její jedenáctiletá dcera Shani Bartová prozradila, že je to trochu zvláštní</td>
</tr>
<tr>
<td>word</td>
<td>Její &lt;unk&gt; dcera &lt;unk&gt; &lt;unk&gt; řekla, že je to trochu divné</td>
</tr>
<tr>
<td>hybrid</td>
<td>Její 11-year-old dcera Shani, řekla, že je to trochu divné</td>
</tr>
<tr>
<td>hybrid</td>
<td>Její &lt;unk&gt; dcera, &lt;unk&gt; &lt;unk&gt;, řekla, že je to &lt;unk&gt; &lt;unk&gt;</td>
</tr>
</tbody>
</table>

• Hybrid: correct, 11-year-old – jedenáctiletá.