Bài 11:
Một số ứng dụng học sâu trong xử lý ngôn ngữ tự nhiên
(Phần 2)

Nội dung

1. Bài toán sinh văn bản: Character-RNN
2. Giới thiệu về bài toán dịch máy
3. Mô hình NMT
4. Cơ chế chú ý (attention)

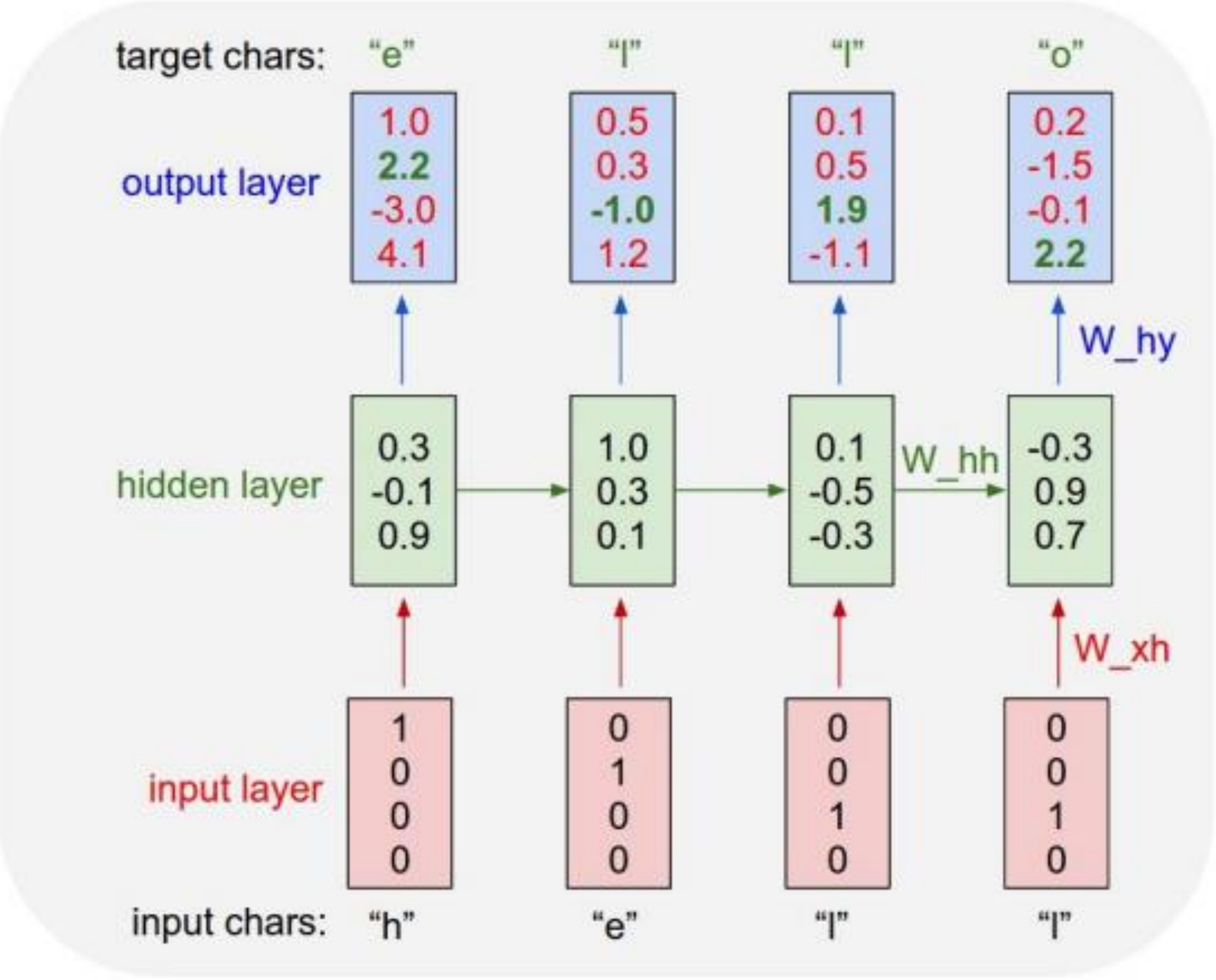
Bài toán sinh văn bản: Character-RNN

Character-RNN

- Từ vựng:

[h, e, l, o]

- Ví dụ huấn luyện mô hình với xâu "hello"



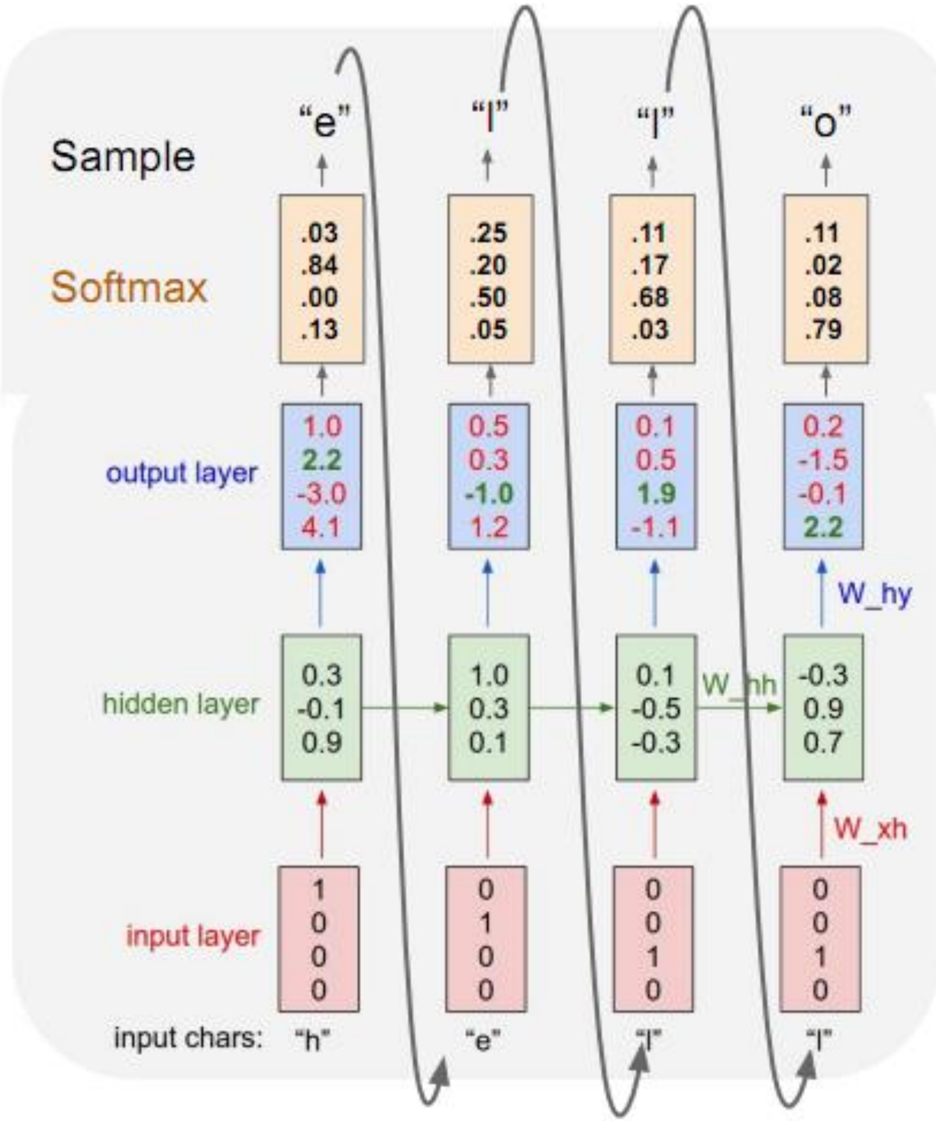
Character-RNN

- Từ vựng:

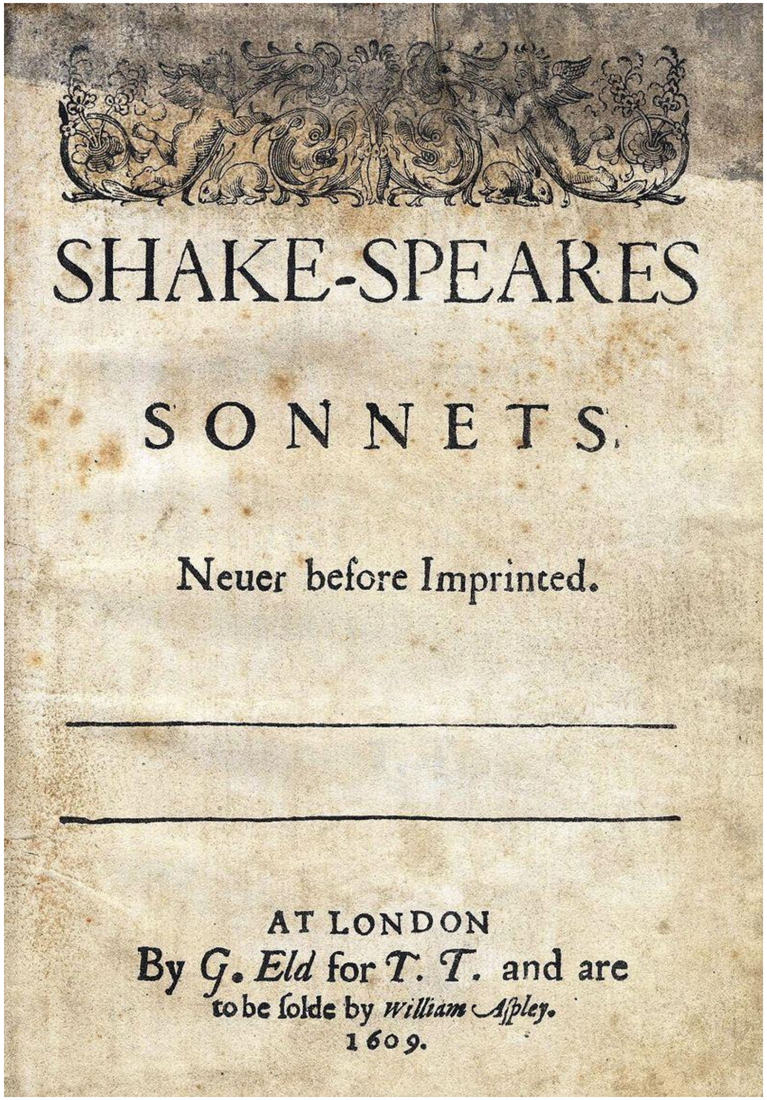
[h, e, l, o]

- Khi suy diễn:

Mỗi lần sinh một ký tự và ký tự này được dùng làm đầu vào cho bước tiếp theo



Character-RNN



SHAKE-SPEARES

SONNETS.

Neuer before Imprinted.

AT LONDON
By G. Eld for T. T. and are
to be solde by William Aspley.
1609.

From fairest creatures we desire increase,
That thereby beauty's rose might never die,
But as the riper should by time decease,
His tender heir might bear his memory:
But thou, contracted to thine own bright eyes,
Feed'st thy light's flame with self-substantial fuel,
Making a famine where abundance lies,
Thyself thy foe, to thy sweet self too cruel:
Thou that art now the world's fresh ornament,
And only herald to the gaudy spring,
Within thine own bud buriest thy content,
And tender churl mak'st waste in niggarding:
  Pity the world, or else this glutton be,
  To eat the world's due, by the grave and thee.

When forty winters shall besiege thy brow,
And dig deep trenches in thy beauty's field,
Thy youth's proud livery so gazed on now,
Will be a tatter'd weed of small worth held:
Then being asked, where all thy beauty lies,
Where all the treasure of thy lusty days;
To say, within thine own deep sunken eyes,
Were an all-eating shame, and thriftless praise.
How much more praise deserv'd thy beauty's use,
If thou couldst answer 'This fair child of mine
Shall sum my count, and make my old excuse,'
Proving his beauty by succession thine!
  This were to be new made when thou art old,
  And see thy blood warm when thou feel'st it cold.

Character-RNN

```
tyntd-iafhatawiaoihrdemot  lytdws  e ,tfti, astai f ogoh eoase rrranbyne 'nhthnee e
plia tklrgd t o idoe ns,smtt   h ne etie h,hregtrs nigtike,aoaenns lng
```

train more

```
"Tmont thithey" fomesscerliund
Keushey. Thom here
sheulke, anmerenith ol sivh I lalterthend Bleipile shuwy fil on aseterlome
coaniogennc Phe lism thond hon at. MeiDimorotion in ther thize."
```

train more

```
Aftair fall unsuch that the hall for Prince Velzonski's that me of
her hearly, and behs to so arwage fiving were to it beloge, pavu say falling misfort
how, and Gogition is so overelical and ofter.
```

train more

```
"Why do what that day," replied Natasha, and wishing to himself the fact the
princess, Princess Mary was easier, fed in had oftened him.
Pierre aking his soul came to the packs and drove up his father-in-law women.
```

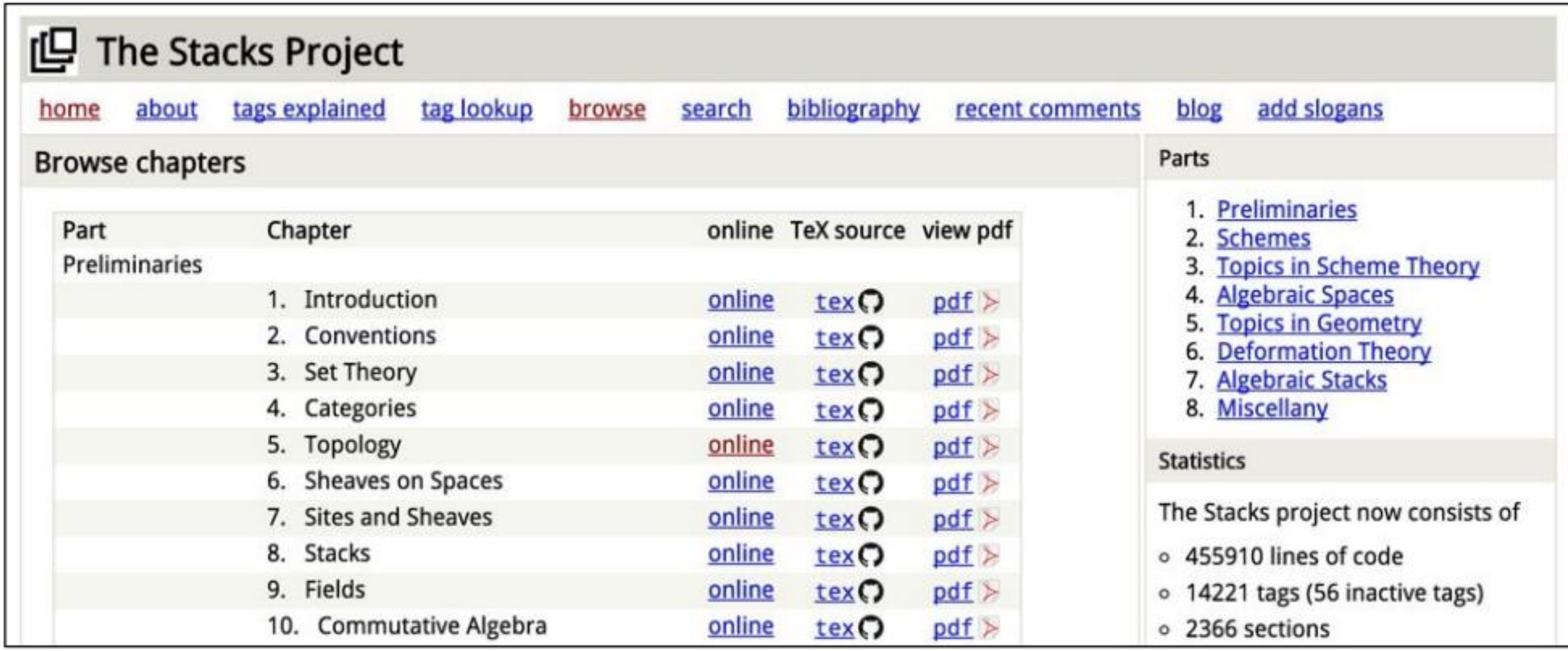
Character-RNN

```
PANDARUS:
Alas, I think he shall be come approached and the day
When little srain would be attain'd into being never fed,
And who is but a chain and subjects of his death,
I should not sleep.

Second Senator:
They are away this miseries, produced upon my soul,
Breaking and strongly should be buried, when I perish
The earth and thoughts of many states.

DUKE VINCENTIO:
Well, your wit is in the care of side and that.

Second Lord:
They would be ruled after this chamber, and
my fair nues begun out of the fact, to be conveyed,
Whose noble souls I'll have the heart of the wars.

Clown:
Come, sir, I will make did behold your worship.

VIOLA:
I'll drink it.
```

```
VIOLA:
Why, Salisbury must find his flesh and thought
That which I am not aps, not a man and in fire,
To show the reining of the raven and the wars
To grace my hand reproach within, and not a fair are hand,
That Caesar and my goodly father's world;
When I was heaven of presence and our fleets,
We spare with hours, but cut thy council I am great,
Murdered and by thy master's ready there
My power to give thee but so much as hell:
Some service in the noble bondman here,
Would show him to her wine.

KING LEAR:
O, if you were a feeble sight, the courtesy of your law,
Your sight and several breath, will wear the gods
With his heads, and my hands are wonder'd at the deeds,
So drop upon your lordship's head, and your opinion
Shall be against your honour.
```

Character-RNN

- The Stacks Project: Sách nguồn mở về hình học đại số



The Stacks Project

home about tags explained tag lookup browse search bibliography recent comments blog add slogans

Browse chapters

Part	Chapter	online	TeX source	view pdf
Preliminaries				
	1. Introduction	online	tex	pdf
	2. Conventions	online	tex	pdf
	3. Set Theory	online	tex	pdf
	4. Categories	online	tex	pdf
	5. Topology	online	tex	pdf
	6. Sheaves on Spaces	online	tex	pdf
	7. Sites and Sheaves	online	tex	pdf
	8. Stacks	online	tex	pdf
	9. Fields	online	tex	pdf
	10. Commutative Algebra	online	tex	pdf

Parts

1. Preliminaries
2. Schemes
3. Topics in Scheme Theory
4. Algebraic Spaces
5. Topics in Geometry
6. Deformation Theory
7. Algebraic Stacks
8. Miscellany

Statistics

The Stacks project now consists of

- 455910 lines of code
- 14221 tags (56 inactive tags)
- 2366 sections

Latex source

http://stacks.math.columbia.edu/

Character-RNN

For $\bigoplus_{n=1,\ldots,m}$ where $\mathcal{L}_{m_\bullet} = 0$, hence we can find a closed subset $\mathcal{H}$ in $\mathcal{H}$ and any sets $\mathcal{F}$ on X, U is a closed immersion of S, then $U \to T$ is a separated algebraic space.

Proof. Proof of (1). It also start we get

$$S = \mathrm{Spec}(R) = U \times_X U \times_X U$$

and the comparicoly in the fibre product covering we have to prove the lemma generated by $\coprod Z \times_U U \to V$. Consider the maps M along the set of points Sch_{fppf} and $U \to U$ is the fibre category of S in U in Section, ?? and the fact that any U affine, see Morphisms, Lemma ??. Hence we obtain a scheme S and any open subset $W \subset U$ in $Sh(G)$ such that $\mathrm{Spec}(R') \to S$ is smooth or an

$$U = \bigcup U_i \times_{S_i} U_i$$

which has a nonzero morphism we may assume that f_i is of finite presentation over S. We claim that $\mathcal{O}_{X,x}$ is a scheme where $x, x', s'' \in S'$ such that $\mathcal{O}_{X,x'} \to \mathcal{O}'_{X',x'}$ is separated. By Algebra, Lemma ?? we can define a map of complexes $\mathrm{GL}_{S'}(x'/S'')$ and we win. □

To prove study we see that $\mathcal{F}|_U$ is a covering of $\mathcal{X}'$, and $\mathcal{T}_i$ is an object of $\mathcal{F}_{X/S}$ for $i > 0$ and $\mathcal{F}_p$ exists and let $\mathcal{F}_i$ be a presheaf of $\mathcal{O}_X$-modules on $\mathcal{C}$ as a $\mathcal{F}$-module. In particular $\mathcal{F} = U/\mathcal{F}$ we have to show that

$$\widetilde{M}^\bullet = \mathcal{I}^\bullet \otimes_{\mathrm{Spec}(k)} \mathcal{O}_{S,s} - i_X^{-1}\mathcal{F})$$

is a unique morphism of algebraic stacks. Note that

$$\mathrm{Arrows} = (Sch/S)^{opp}_{fppf}, (Sch/S)_{fppf}$$

and

$$V = \Gamma(S, \mathcal{O}) \longmapsto (U, \mathrm{Spec}(A))$$

is an open subset of X. Thus U is affine. This is a continuous map of X is the inverse, the groupoid scheme S.

Proof. See discussion of sheaves of sets. □

The result for prove any open covering follows from the less of Example ??. It may replace S by $X_{spaces,\acute{e}tale}$ which gives an open subspace of X and T equal to S_{Zar}, see Descent, Lemma ??. Namely, by Lemma ?? we see that R is geometrically regular over S.

Lemma 0.1. *Assume (3) and (3) by the construction in the description.*

Suppose $X = \lim |X|$ (by the formal open covering X and a single map $\underline{Proj}_X(\mathcal{A}) = \mathrm{Spec}(B)$ over U compatible with the complex

$$Set(\mathcal{A}) = \Gamma(X, \mathcal{O}_{X,\mathcal{O}_X}).$$

When in this case of to show that $\mathcal{Q} \to \mathcal{C}_{Z/X}$ is stable under the following result in the second conditions of (1), and (3). This finishes the proof. By Definition ?? (without element is when the closed subschemes are catenary. If T is surjective we may assume that T is connected with residue fields of S. Moreover there exists a closed subspace $Z \subset X$ of X where U in X' is proper (some defining as a closed subset of the uniqueness it suffices to check the fact that the following theorem

(1) *f is locally of finite type. Since $S = \mathrm{Spec}(R)$ and $Y = \mathrm{Spec}(R)$.*

Proof. This is form all sheaves of sheaves on X. But given a scheme U and a surjective étale morphism $U \to X$. Let $U \cap U = \coprod_{i=1,\ldots,n} U_i$ be the scheme X over S at the schemes $X_i \to X$ and $U = \lim_i X_i$. □

The following lemma surjective restrocomposes of this implies that $\mathcal{F}_{x_0} = \mathcal{F}_{x_0} = \mathcal{F}_{X,\ldots,0}$.

Lemma 0.2. *Let X be a locally Noetherian scheme over S, $E = \mathcal{F}_{X/S}$. Set $\mathcal{I} = \mathcal{J}_1 \subset \mathcal{I}'_n$. Since $\mathcal{I}^n \subset \mathcal{I}^n$ are nonzero over $i_0 \leq \mathfrak{p}$ is a subset of $\mathcal{J}_{n,0} \circ \overline{A}_2$ works.*

Lemma 0.3. *In Situation ??. Hence we may assume $\mathfrak{q}' = 0$.*

Proof. We will use the property we see that $\mathfrak{p}$ is the mext functor (??). On the other hand, by Lemma ?? we see that

$$D(\mathcal{O}_{X'}) = \mathcal{O}_X(D)$$

where K is an F-algebra where δ_{n+1} is a scheme over S. □

Character-RNN

Proof. Omitted. □

Lemma 0.1. *Let $\mathcal{C}$ be a set of the construction.*

Let $\mathcal{C}$ be a gerber covering. Let $\mathcal{F}$ be a quasi-coherent sheaves of $\mathcal{O}$-modules. We have to show that

$$\mathcal{O}_{\mathcal{O}_X} = \mathcal{O}_X(\mathcal{L})$$

.

Proof. This is an algebraic space with the composition of sheaves $\mathcal{F}$ on $X_{\acute{e}tale}$ we have

$$\mathcal{O}_X(\mathcal{F}) = \{morph_1 \times_{\mathcal{O}_X} (\mathcal{G}, \mathcal{F})\}$$

where $\mathcal{G}$ defines an isomorphism $\mathcal{F} \to \mathcal{F}$ of $\mathcal{O}$-modules. □

Lemma 0.2. *This is an integer $\mathcal{Z}$ is injective.*

Proof. See Spaces, Lemma ??. □

Lemma 0.3. *Let S be a scheme. Let X be a scheme and X is an affine open covering. Let $\mathcal{U} \subset \mathcal{X}$ be a canonical and locally of finite type. Let X be a scheme. Let X be a scheme which is equal to the formal complex.*

The following to the construction of the lemma follows.

Let X be a scheme. Let X be a scheme covering. Let

$$b : X \to Y' \to Y \to Y \to Y' \times_X Y \to X.$$

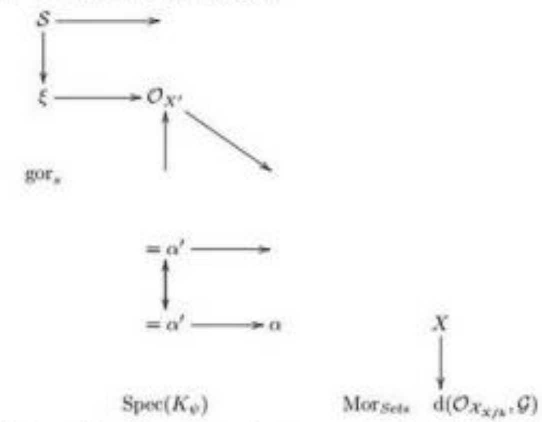
be a morphism of algebraic spaces over S and Y.

Proof. Let X be a nonzero scheme of X. Let X be an algebraic space. Let $\mathcal{F}$ be a quasi-coherent sheaf of $\mathcal{O}_X$-modules. The following are equivalent

(1) $\mathcal{F}$ is an algebraic space over S.
(2) If X is an affine open covering.

Consider a common structure on X and X the functor $\mathcal{O}_X(U)$ which is locally of finite type. □

This since $\mathcal{F} \in \mathcal{F}$ and $x \in \mathcal{G}$ the diagram



is a limit. Then $\mathcal{G}$ is a finite type and assume S is a flat and $\mathcal{F}$ and $\mathcal{G}$ is a finite type f_*. This is of finite type diagrams, and

- the composition of $\mathcal{G}$ is a regular sequence,
- $\mathcal{O}_{X'}$ is a sheaf of rings.

□

Proof. We have see that $X = \mathrm{Spec}(R)$ and $\mathcal{F}$ is a finite type representable by algebraic space. The property $\mathcal{F}$ is a finite morphism of algebraic stacks. Then the cohomology of X is an open neighbourhood of U. □

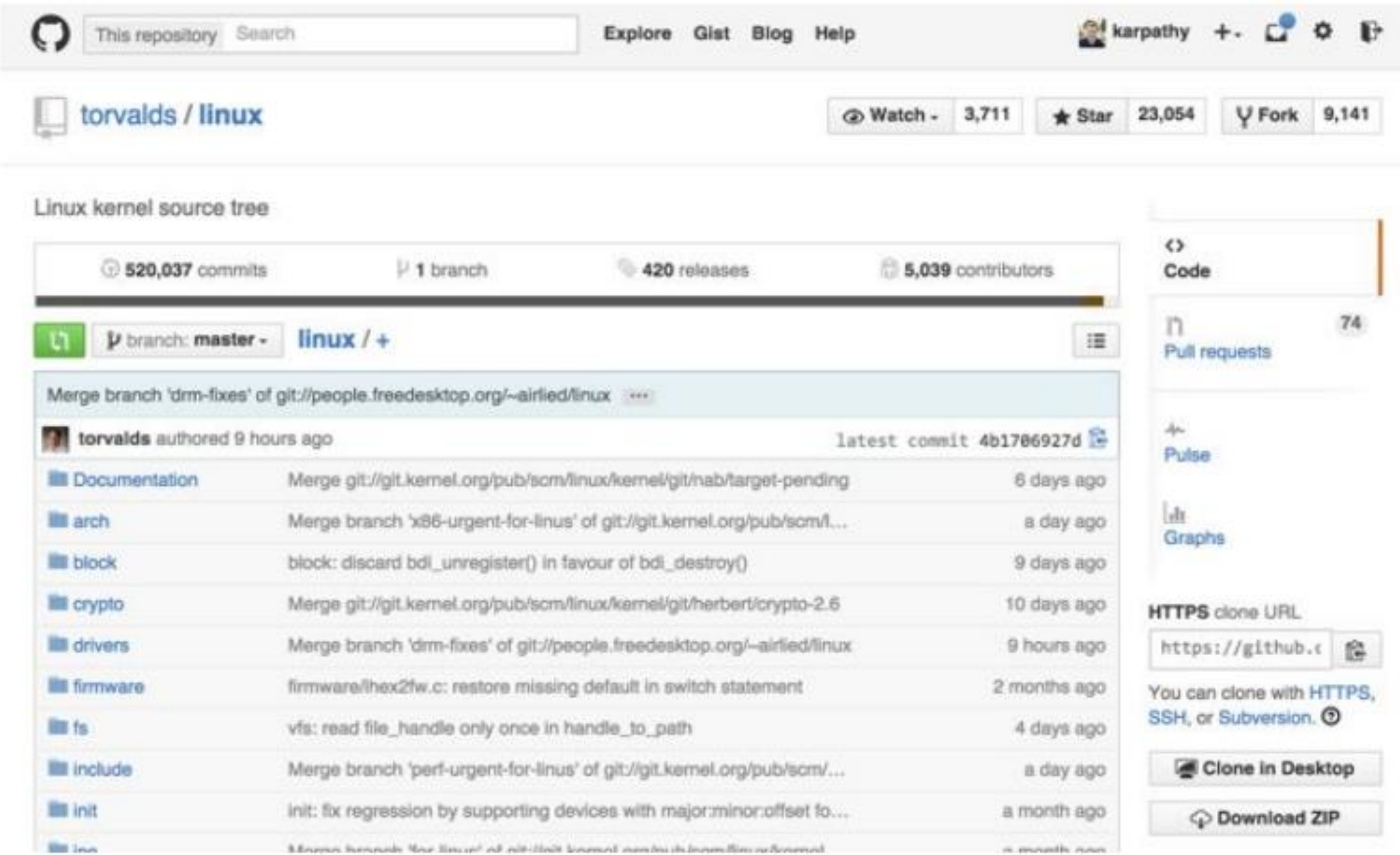
Proof. This is clear that $\mathcal{G}$ is a finite presentation, see Lemmas ??.
A *reduced above* we conclude that U is an open covering of $\mathcal{C}$. The functor $\mathcal{F}$ is a "field

$$\mathcal{O}_{X,x} \longrightarrow \mathcal{F}_{\bar{x}} \ \ \text{-}1(\mathcal{O}_{X_{\acute{e}tale}}) \longrightarrow \mathcal{O}^{-1}_{X_\ell}\mathcal{O}_{X_\lambda}(\mathcal{O}^{\overline{v}}_{X_\eta})$$

is an isomorphism of covering of $\mathcal{O}_{X_i}$. If $\mathcal{F}$ is the unique element of $\mathcal{F}$ such that X is an isomorphism.
The property $\mathcal{F}$ is a disjoint union of Proposition ?? and we can filtered set of presentations of a scheme $\mathcal{O}_X$-algebra with $\mathcal{F}$ are opens of finite type over S.
If $\mathcal{F}$ is a scheme theoretic image points. □

If $\mathcal{F}$ is a finite direct sum $\mathcal{O}_{X_\lambda}$ is a closed immersion, see Lemma ??. This is a sequence of $\mathcal{F}$ is a similar morphism.

Character-RNN



Character-RNN

```c
static void do_command(struct seq_file *m, void *v)
{
  int column = 32 << (cmd[2] & 0x80);
  if (state)
    cmd = (int)(int_state ^ (in_8(&ch->ch_flags) & Cmd) ? 2 : 1);
  else
    seq = 1;
  for (i = 0; i < 16; i++) {
    if (k & (1 << 1))
      pipe = (in_use & UMXTHREAD_UNCCA) +
        ((count & 0x00000000fffffff8) & 0x000000f) << 8;
    if (count == 0)
      sub(pid, ppc_md.kexec_handle, 0x20000000);
    pipe_set_bytes(i, 0);
  }
  /* Free our user pages pointer to place camera if all dash */
  subsystem_info = &of_changes[PAGE_SIZE];
  rek_controls(offset, idx, &soffset);
  /* Now we want to deliberately put it to device */
  control_check_polarity(&context, val, 0);
  for (i = 0; i < COUNTER; i++)
    seq_puts(s, "policy ");
}
```

Generated C code

Character-RNN

```
/*
 *  Copyright (c) 2006-2010, Intel Mobile Communications.  All rights reserved.
 *
 *   This program is free software; you can redistribute it and/or modify it
 * under the terms of the GNU General Public License version 2 as published by
 * the Free Software Foundation.
 *
 *        This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 *
 *  GNU General Public License for more details.
 *
 *   You should have received a copy of the GNU General Public License
 *    along with this program; if not, write to the Free Software Foundation,
 *  Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
 */

#include <linux/kexec.h>
#include <linux/errno.h>
#include <linux/io.h>
#include <linux/platform_device.h>
#include <linux/multi.h>
#include <linux/ckevent.h>

#include <asm/io.h>
#include <asm/prom.h>
#include <asm/e820.h>
#include <asm/system_info.h>
#include <asm/setew.h>
#include <asm/pgproto.h>
```

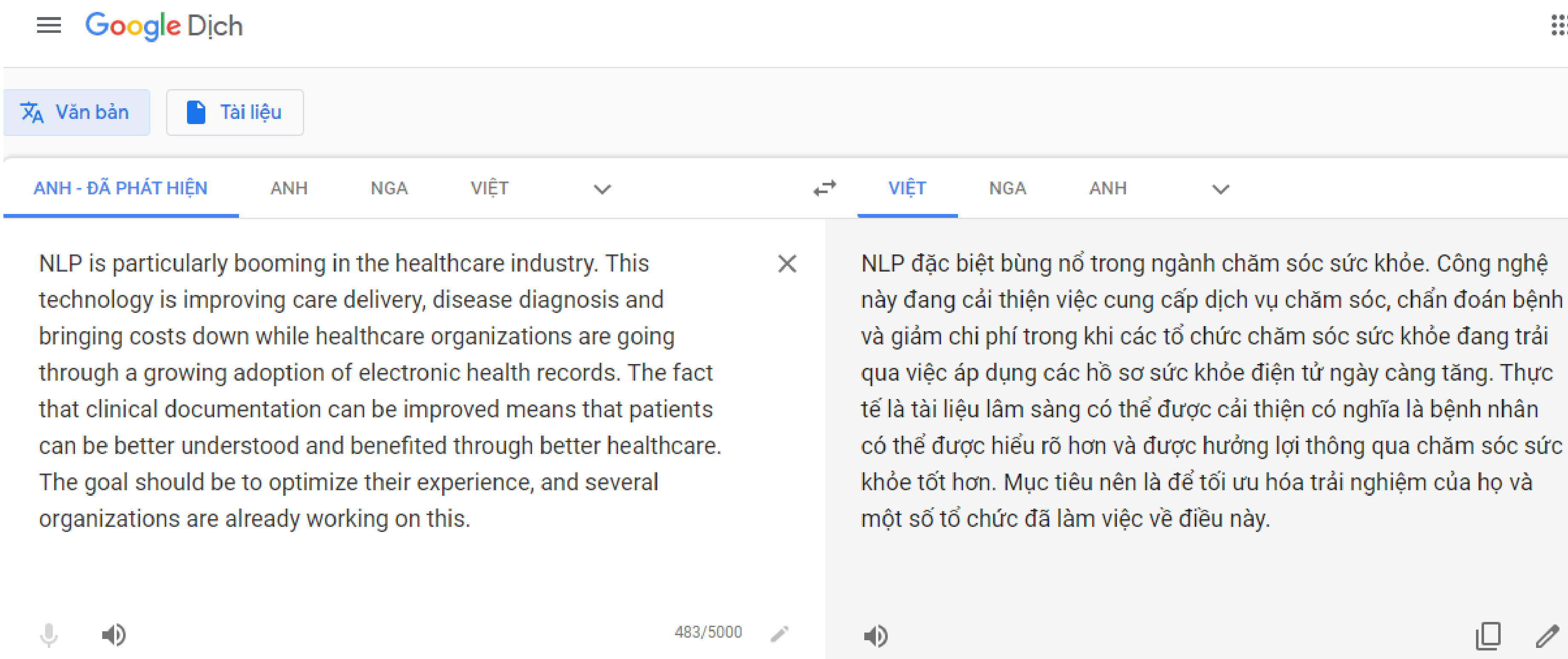
Character-RNN

```
#include <asm/io.h>
#include <asm/prom.h>
#include <asm/e820.h>
#include <asm/system_info.h>
#include <asm/setew.h>
#include <asm/pgproto.h>

#define REG_PG    vesa_slot_addr_pack
#define PFM_NOCOMP  AFSR(0, load)
#define STACK_DDR(type)     (func)

#define SWAP_ALLOCATE(nr)     (e)
#define emulate_sigs()  arch_get_unaligned_child()
#define access_rw(TST)  asm volatile("movd %%esp, %0, %3" : : "r" (0));   \
  if (__type & DO_READ)

static void stat_PC_SEC __read_mostly offsetof(struct seq_argsqueue, \
          pC>[1]);

static void
os_prefix(unsigned long sys)
{
#ifdef CONFIG_PREEMPT
  PUT_PARAM_RAID(2, sel) = get_state_state();
  set_pid_sum((unsigned long)state, current_state_str(),
           (unsigned long)-1->lr_full; low;
}
```

Giới thiệu về bài toán dịch máy

Dịch máy

- Google translate



Dịch máy – Machine Translation

- **Dịch máy (MT)** là thao tác dịch một câu x từ một ngôn ngữ (gọi là ngôn ngữ nguồn) sang một câu y trong ngôn ngữ khác (gọi là ngôn ngữ đích)

x: *L'homme est né libre, et partout il est dans les fers*

↓

y: *Man is born free, but everywhere he is in chains*

Dịch máy – Machine Translation

- Bắt đầu từ những năm 1950
- Dịch từ Nga sang Anh (nhu cầu xuất phát từ chiến tranh lạnh)
- Hệ thống dịch chủ yếu theo quy tắc (rule-based), dùng từ điển để ánh xạ các từ tiếng Nga sang tiếng Anh

1 minute video showing 1954 MT: https://youtu.be/K-HfpsHPmvw

Dịch máy – Machine Translation

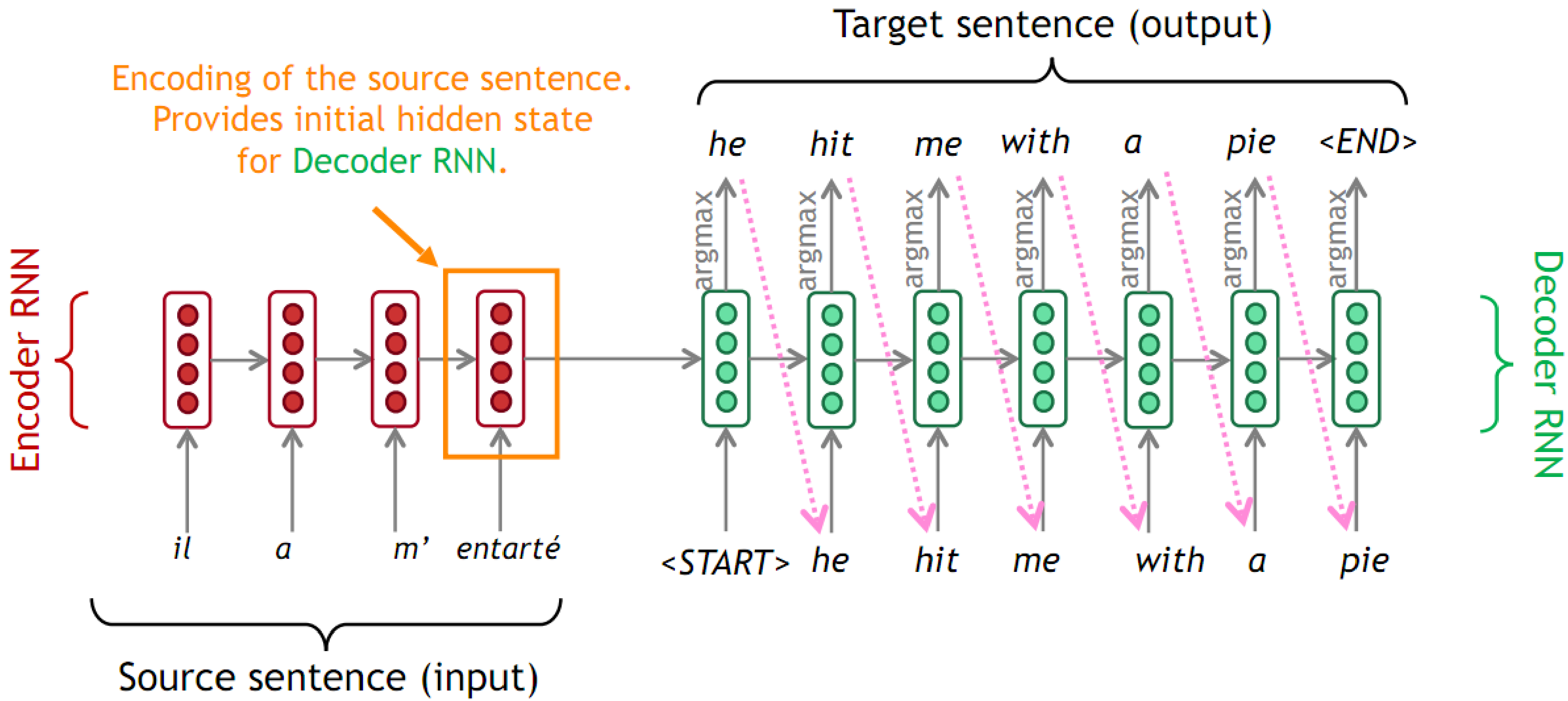
- 1990s-2010s: Statistical Machine Translation (**SMT**)
- Dùng suy diễn Bayes:

$$\mathrm{argmax}_y P(y|x) = \mathrm{argmax}_y P(x|y)P(y)$$

- Các hệ thống tốt nhất theo hướng tiếp cận này rất phức tạp, mỗi hệ thống chứa nhiều mô-đun nhỏ được thiết kế độc lập nhau
- Cần nhiều xử lý thủ công và sức người
- Khi chuyển sang cặp ngôn ngữ khác phải làm lại thủ công từ đầu, không tái sử dụng được sức người

Mô hình NMT
(Neural Machine Translation)

Mô hình sequence-to-sequence



- Encoder RNN sinh ra "thông tin mã hóa" (encoding) của câu nguồn
- Decoder RNN sinh ra câu đích dựa trên thông tin mã hóa của câu nguồn

Mô hình sequence-to-sequence

- Mô hình seq2seq có thể sử dụng cho nhiều bài toán khác như:
  - Tóm lược văn bản (văn bản dài → văn bản ngắn)
  - Hội thoại (câu nói trước → câu nói tiếp theo)
  - Sinh code (ngôn ngữ tự nhiên → code python)
  - …

Mô hình sequence-to-sequence

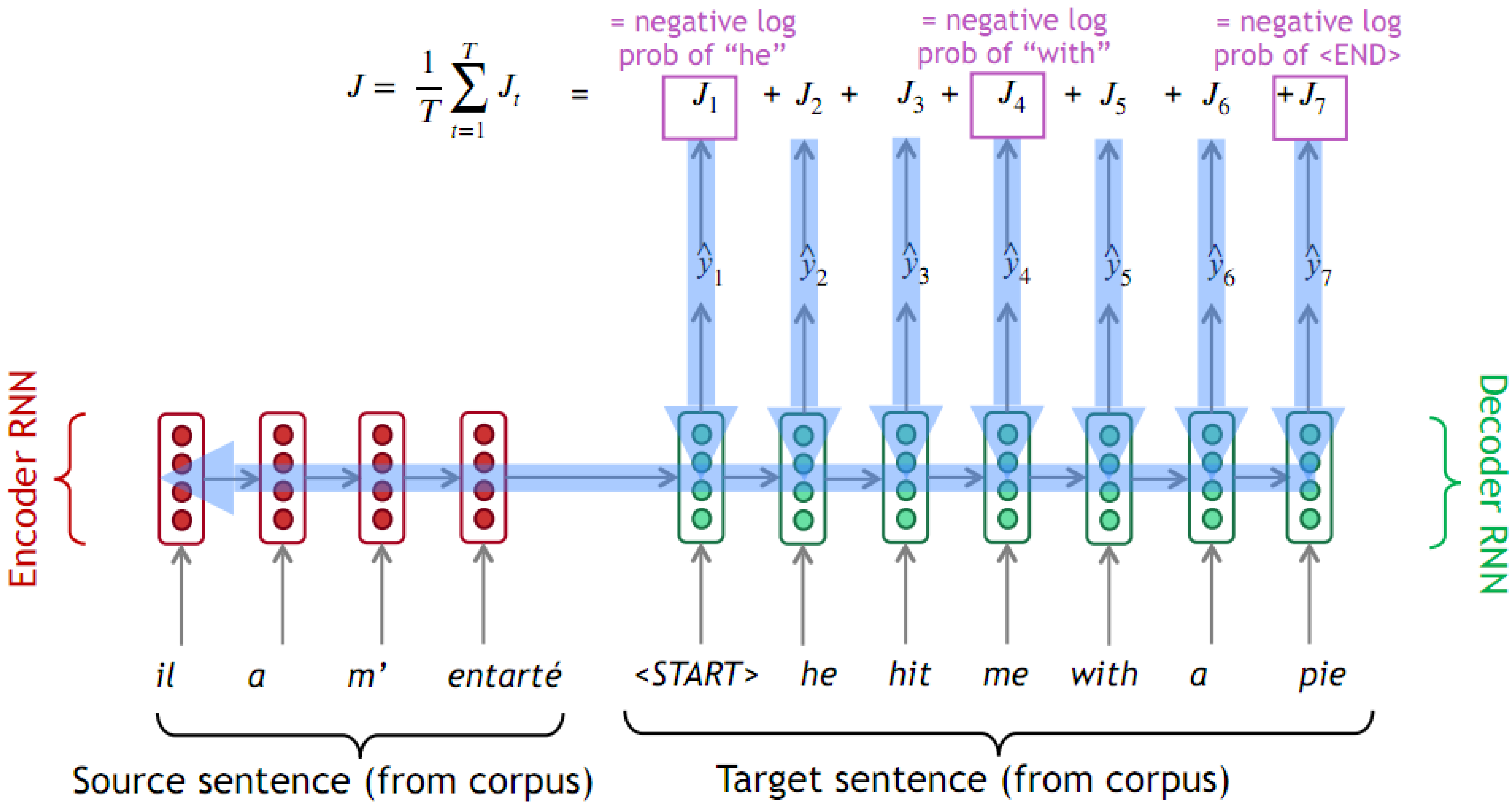
- NMT tính trực tiếp $P(y|x)$:

$$P(y|x) = P(y_1|x)\,P(y_2|y_1,x)\,P(y_3|y_1,y_2,x)\ldots P(y_T|y_1,\ldots,y_{T-1},x)$$

- Làm thế nào để huấn luyện NMT?
- **Trả lời:** Sử dụng một corpus lớn các cặp dịch thuật song ngữ.

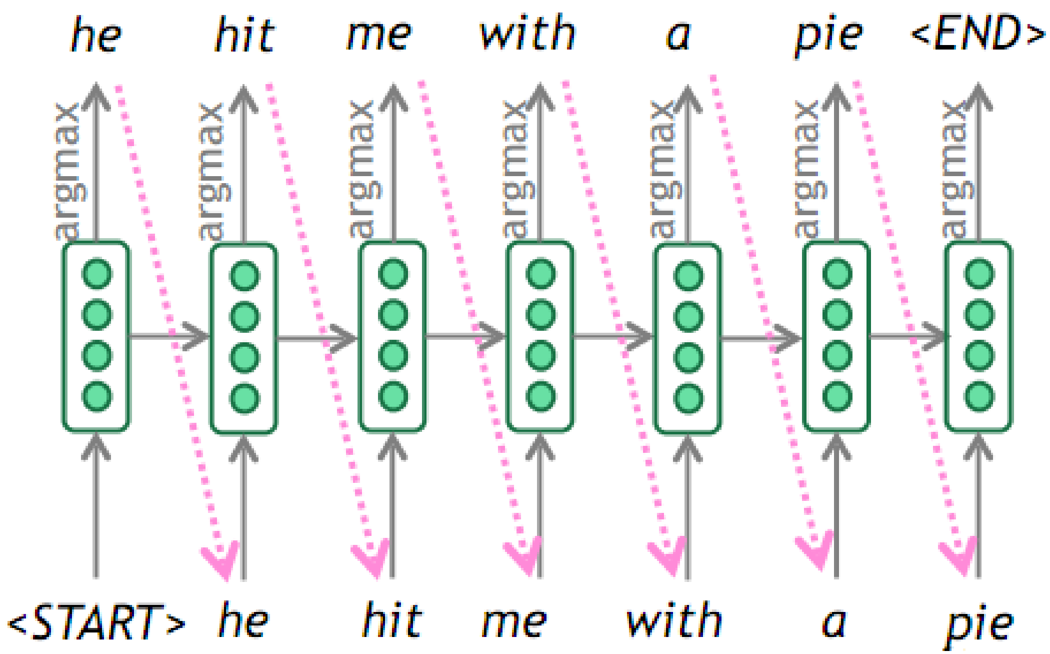
Mô hình sequence-to-sequence

- Huấn luyện NMT



Mô hình sequence-to-sequence

- Giải mã ra câu đích bằng cách lấy argmax tại từng bước
- Đây là cách giải mã tham lam
- Nếu lỡ sai ở một bước nào đó là sẽ sai luôn các bước sau, không có cách nào quay lại để sửa.



Phương pháp duyệt

- Ta mong muốn tìm được câu đích y (độ dài T) cực đại hóa xác suất hậu nghiệm:

$$
\begin{aligned}
P(y|x) &= P(y_1|x)\,P(y_2|y_1,x)\,P(y_3|y_1,y_2,x)\ldots,P(y_T|y_1,\ldots,y_{T-1},x) \\
&= \prod_{t=1}^{T} P(y_t|y_1,\ldots,y_{t-1},x)
\end{aligned}
$$

- Ta có thể tính với tất cả các phương án của y.
- Độ phức tạp V^T với V là kích thước tập từ vựng.

Tìm kiếm chùm – beam search

- Ý tưởng: Tại mỗi bước giải mã, ta duy trì *k* phương án bộ phận có xác suất xảy ra cao nhất (gọi là các giả thuyết)
- *k* là kích thước chùm (beam size)
- Một giả thuyết $y_1, y_2, \ldots, y_t$ có điểm bằng log giá trị xác suất của nó:

$$\text{score}(y_1, \ldots, y_t) = \log P_{\text{LM}}(y_1, \ldots, y_t | x) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$$

- Tất cả điểm score đều âm, điểm càng cao càng tốt
- Ta sẽ giữ lại *k* giả thuyết có điểm score cao nhất tại mỗi bước
- Tìm kiếm chùm không đảm bảo tìm được lời giải tối ưu
- Nhưng hiệu quả hơn rất nhiều so với phương pháp duyệt

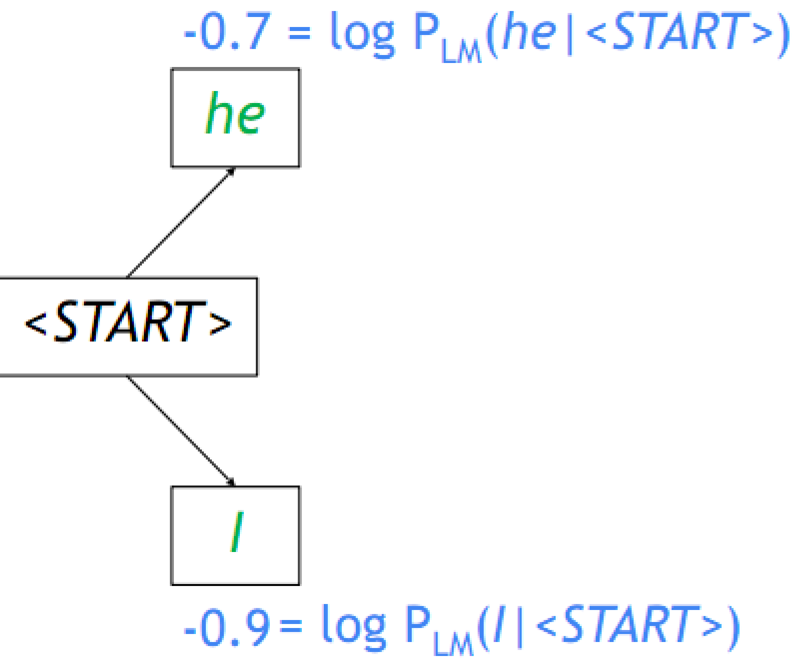
Ví dụ beam search với k = 2

- Tính toán phân phối xác suất từ tiếp theo

<START>

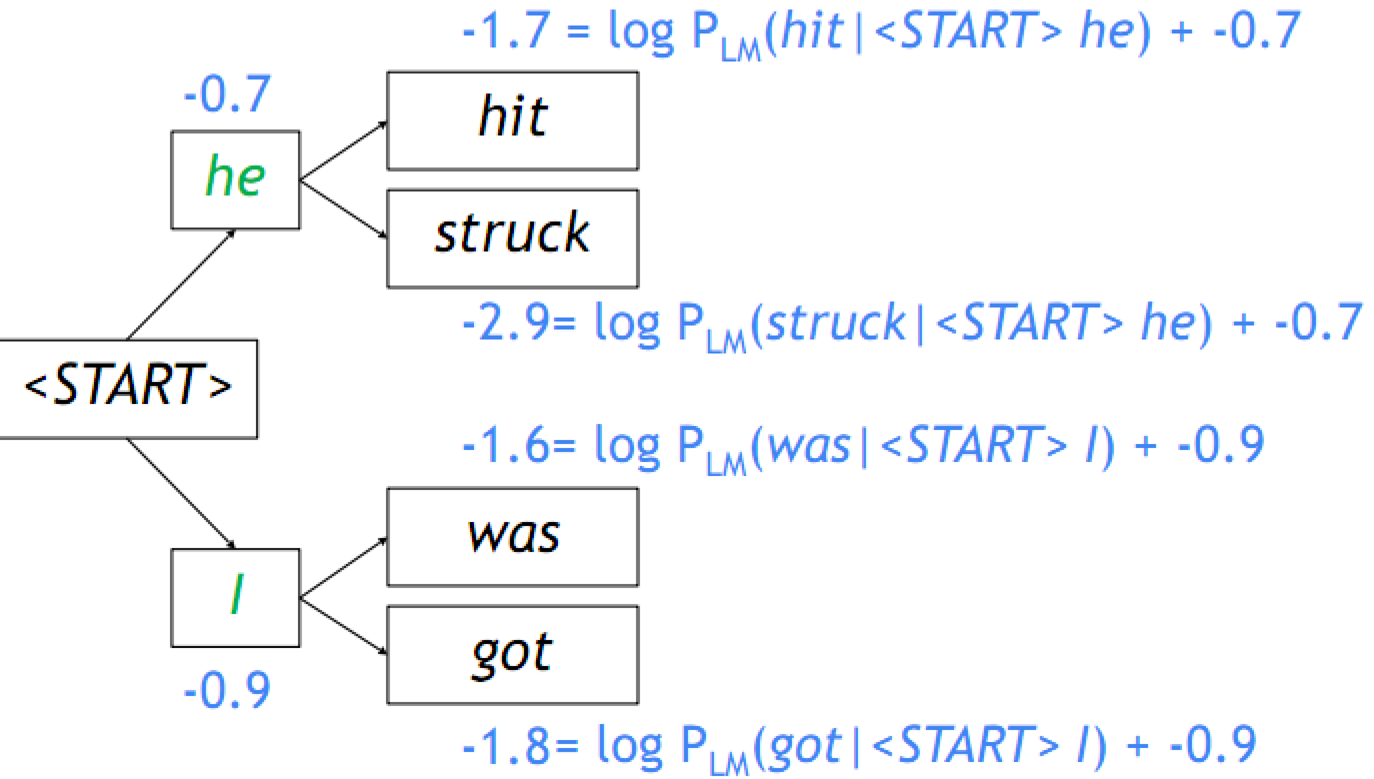
Ví dụ beam search với k = 2

- Giữ hai phương án với điểm cao nhất



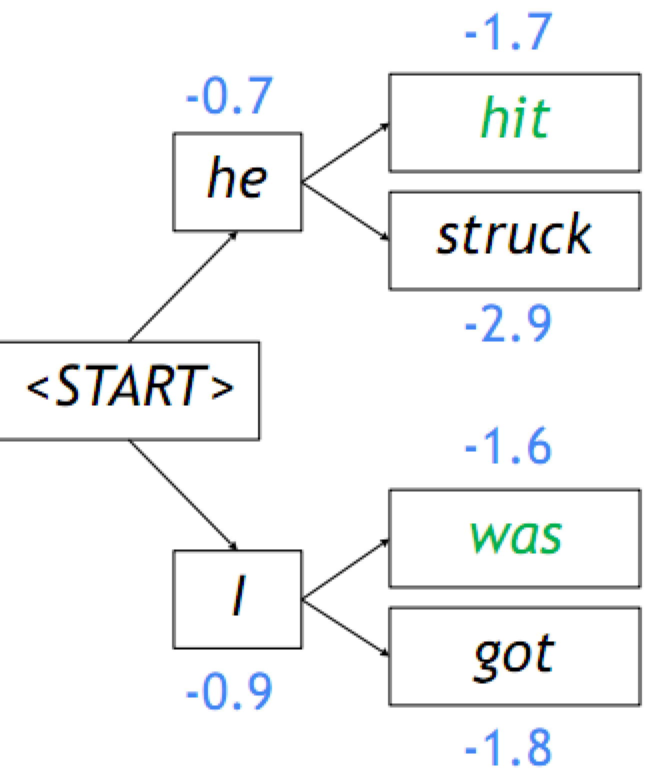
Ví dụ beam search với k = 2

- Với mỗi giả thuyết tìm tiếp *k* giả thuyết tiếp theo có điểm cao nhất



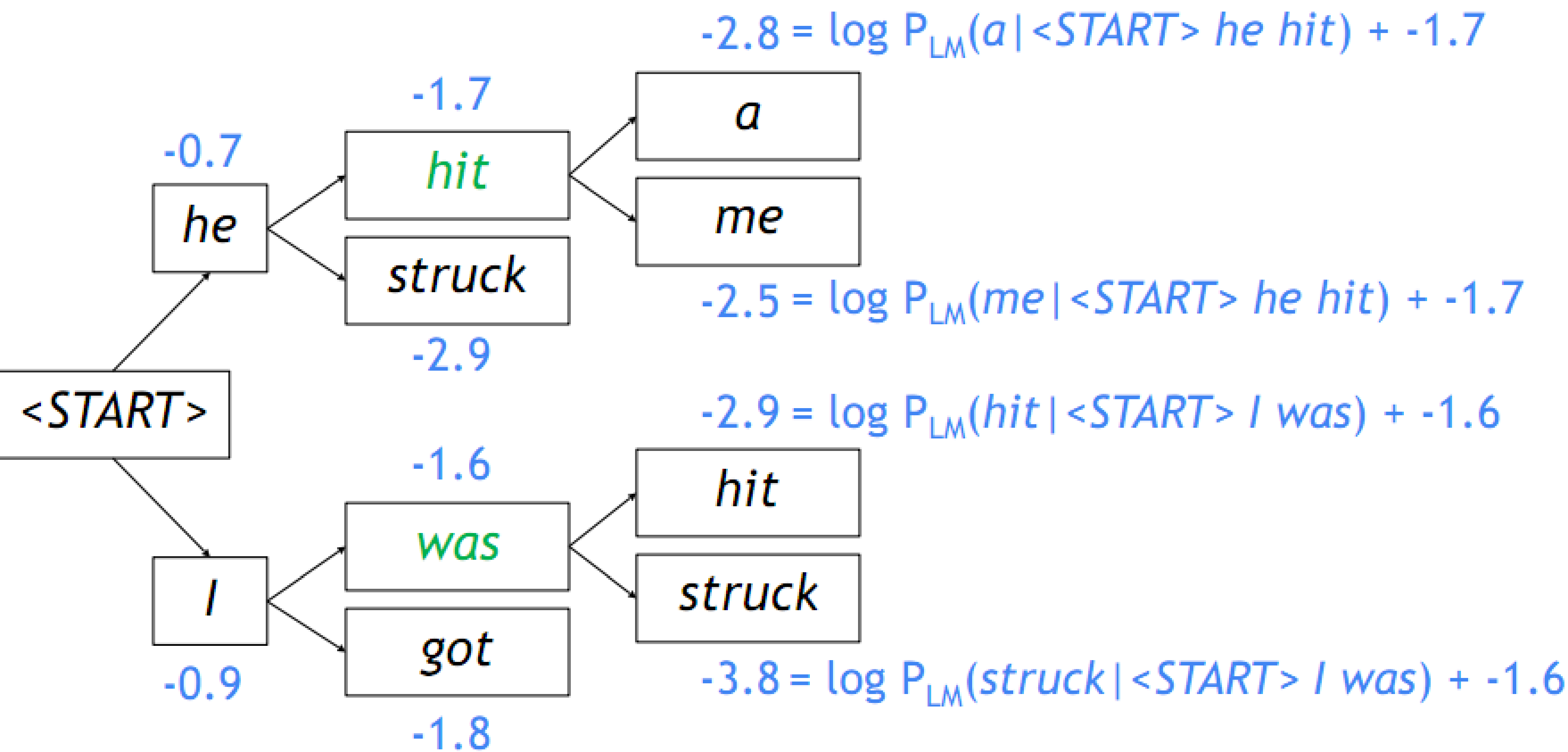
Ví dụ beam search với k = 2

- Trong k^2 giả thuyết mới ta chỉ giữ lại k giả thuyết điểm cao nhất



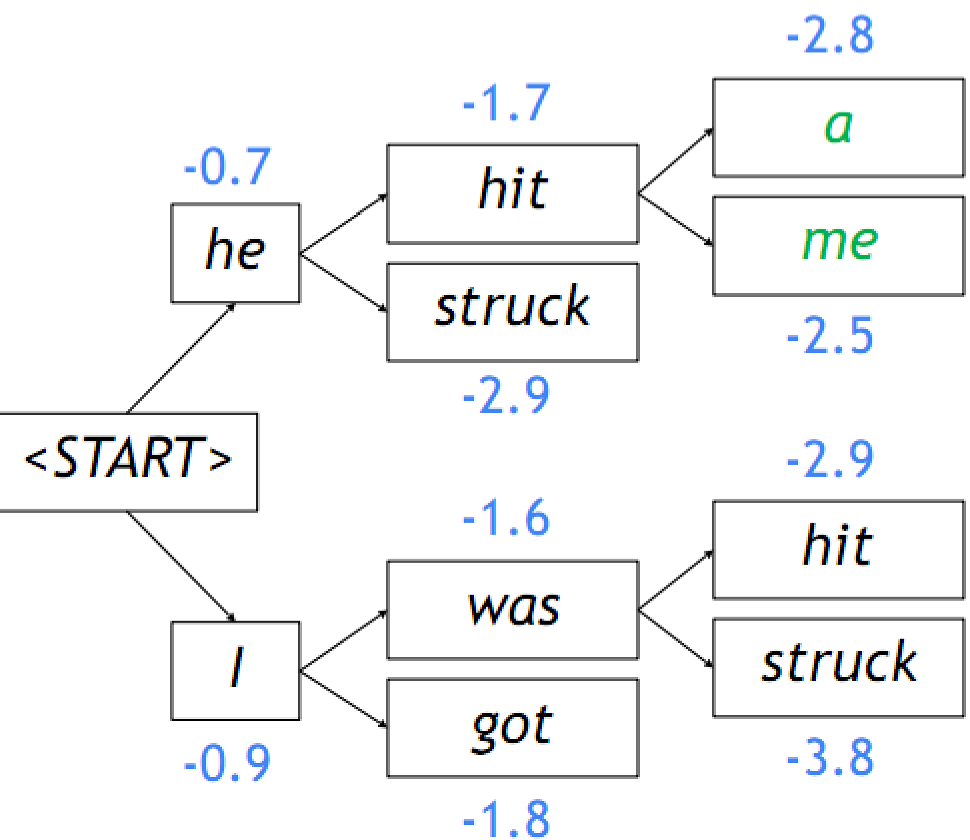
Ví dụ beam search với k = 2

- Với mỗi giả thuyết tìm tiếp *k* giả thuyết tiếp theo có điểm cao nhất



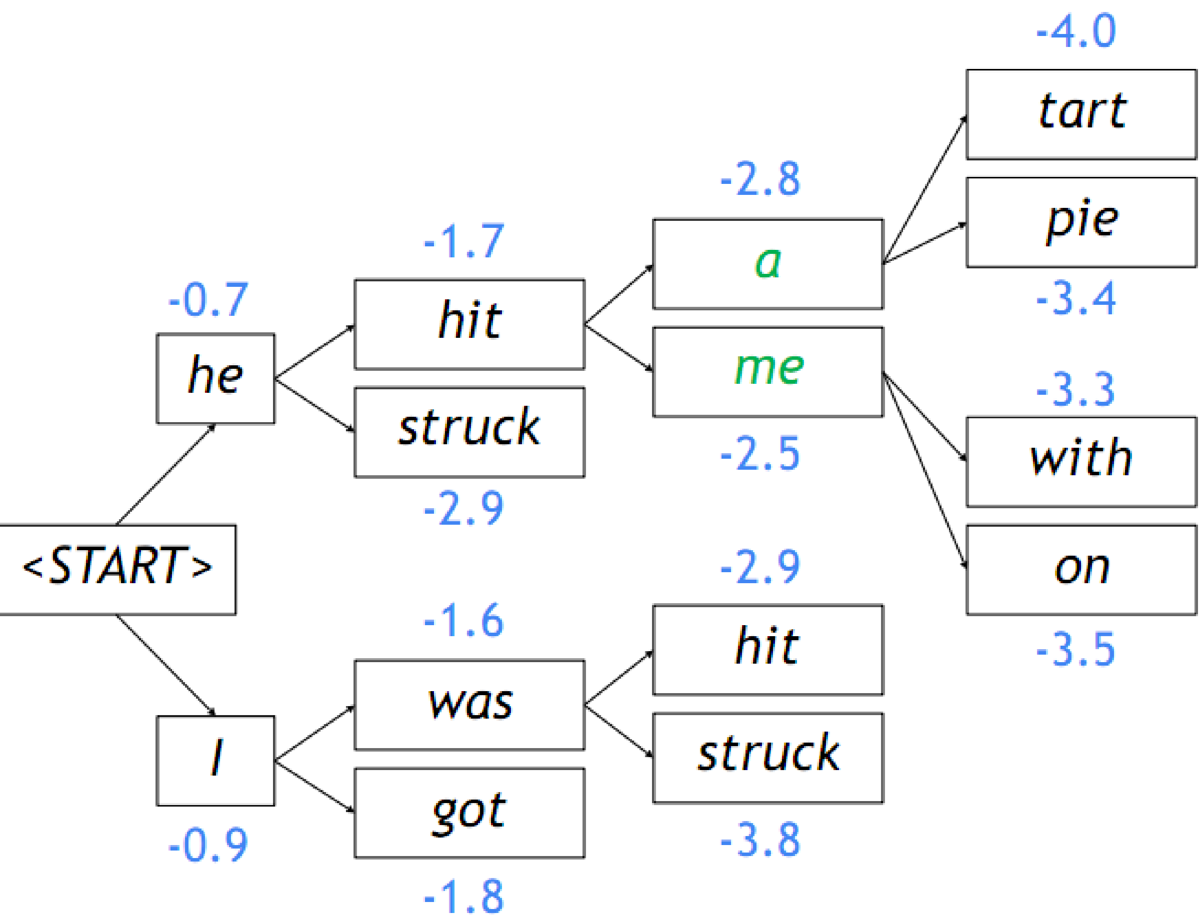
Ví dụ beam search với k = 2

- Trong k^2 giả thuyết mới ta chỉ giữ lại *k* giả thuyết điểm cao nhất



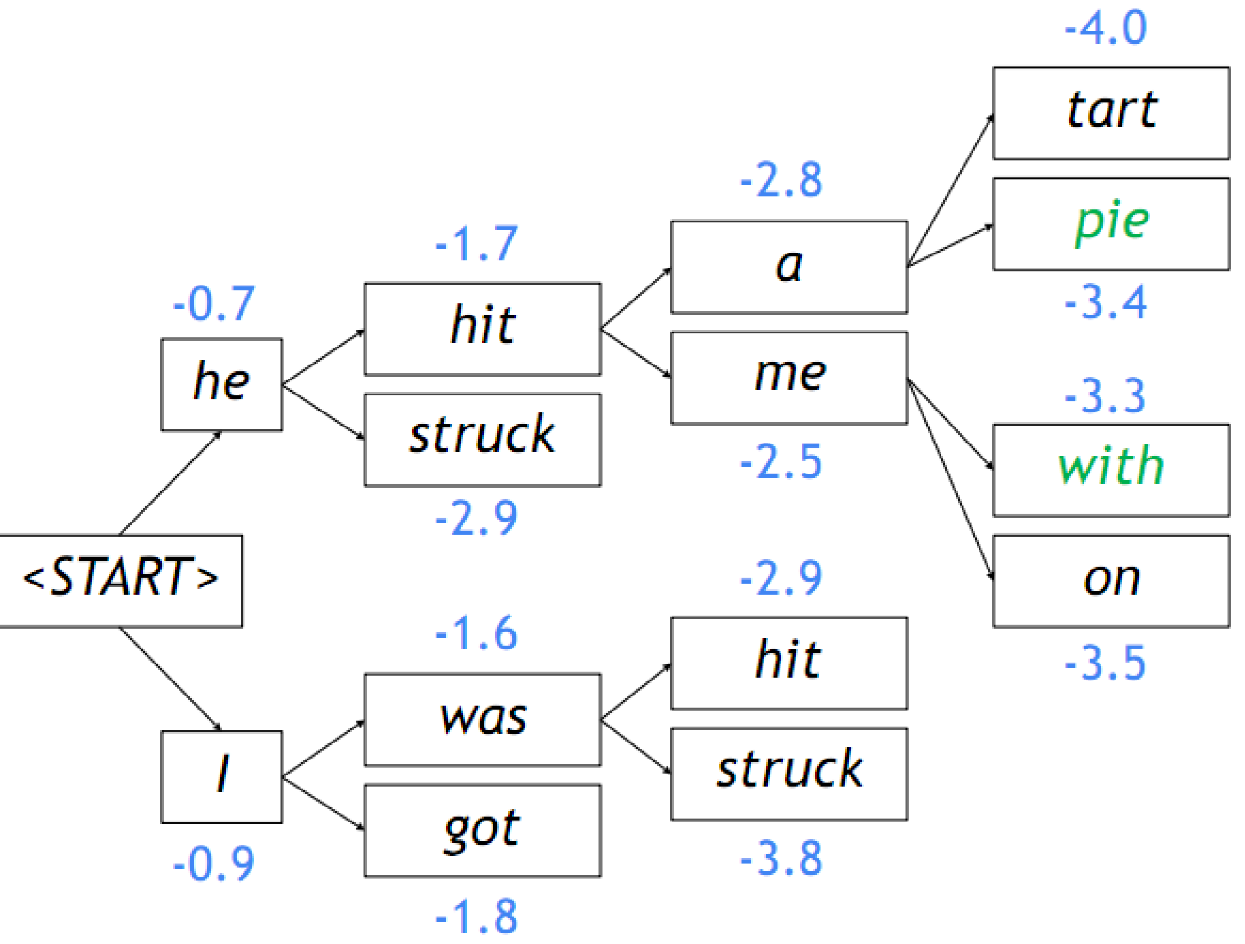
Ví dụ beam search với k = 2

- Với mỗi giả thuyết tìm tiếp *k* giả thuyết tiếp theo có điểm cao nhất



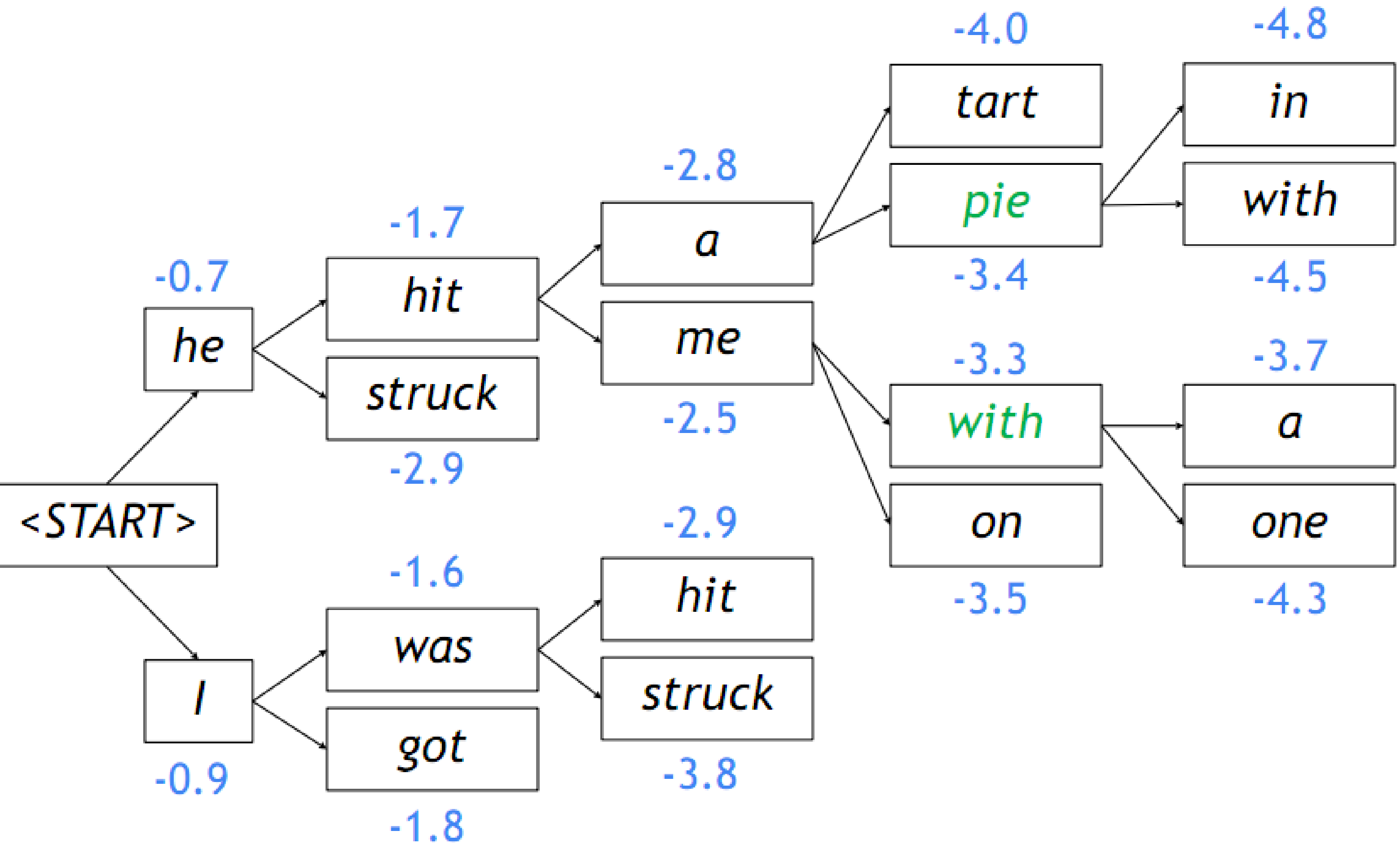
Ví dụ beam search với k = 2

- Trong k^2 giả thuyết mới ta chỉ giữ lại k giả thuyết điểm cao nhất



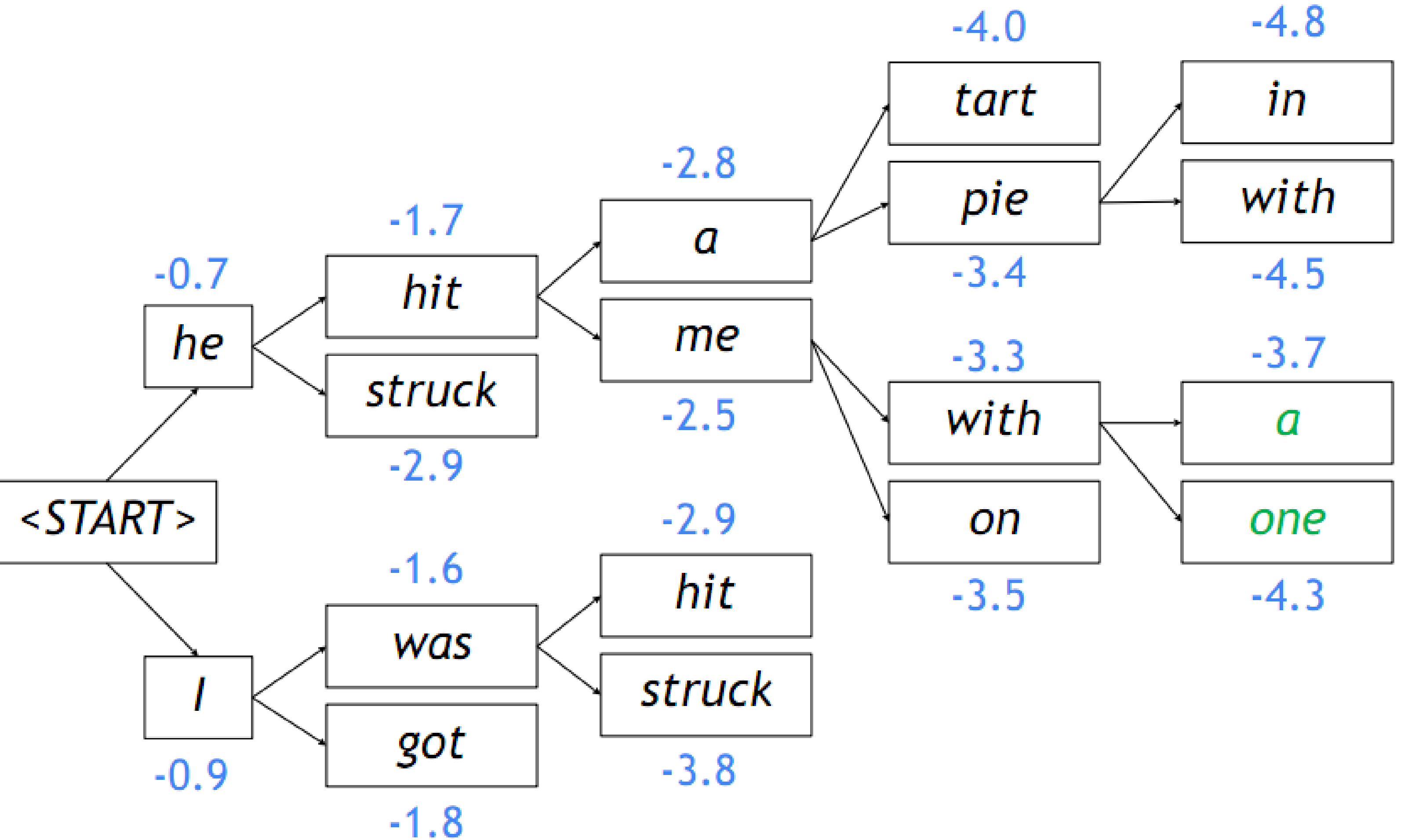
Ví dụ beam search với k = 2

- Với mỗi giả thuyết tìm tiếp *k* giả thuyết tiếp theo có điểm cao nhất



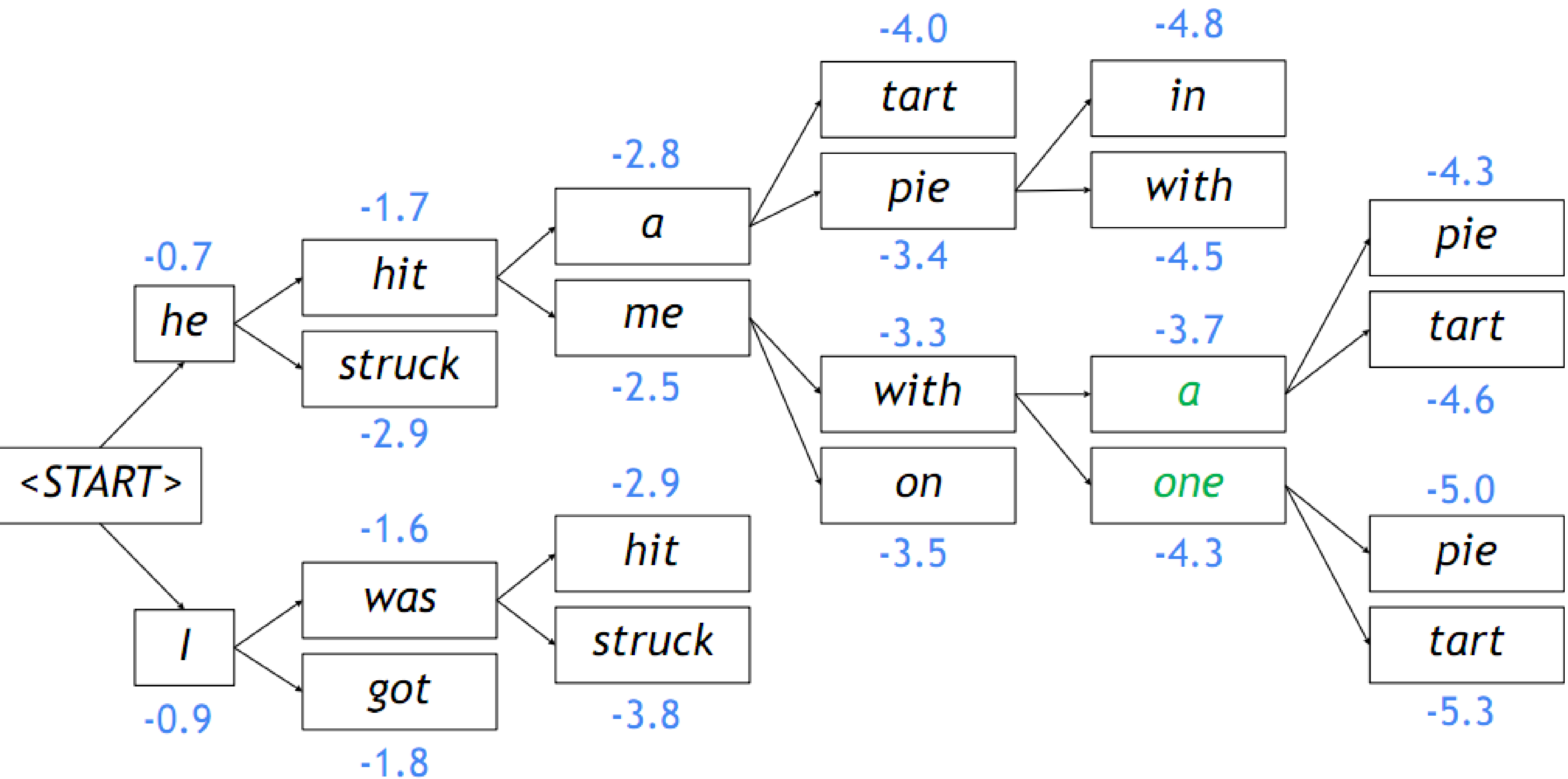
Ví dụ beam search với k = 2

- Trong k^2 giả thuyết mới ta chỉ giữ lại k giả thuyết điểm cao nhất



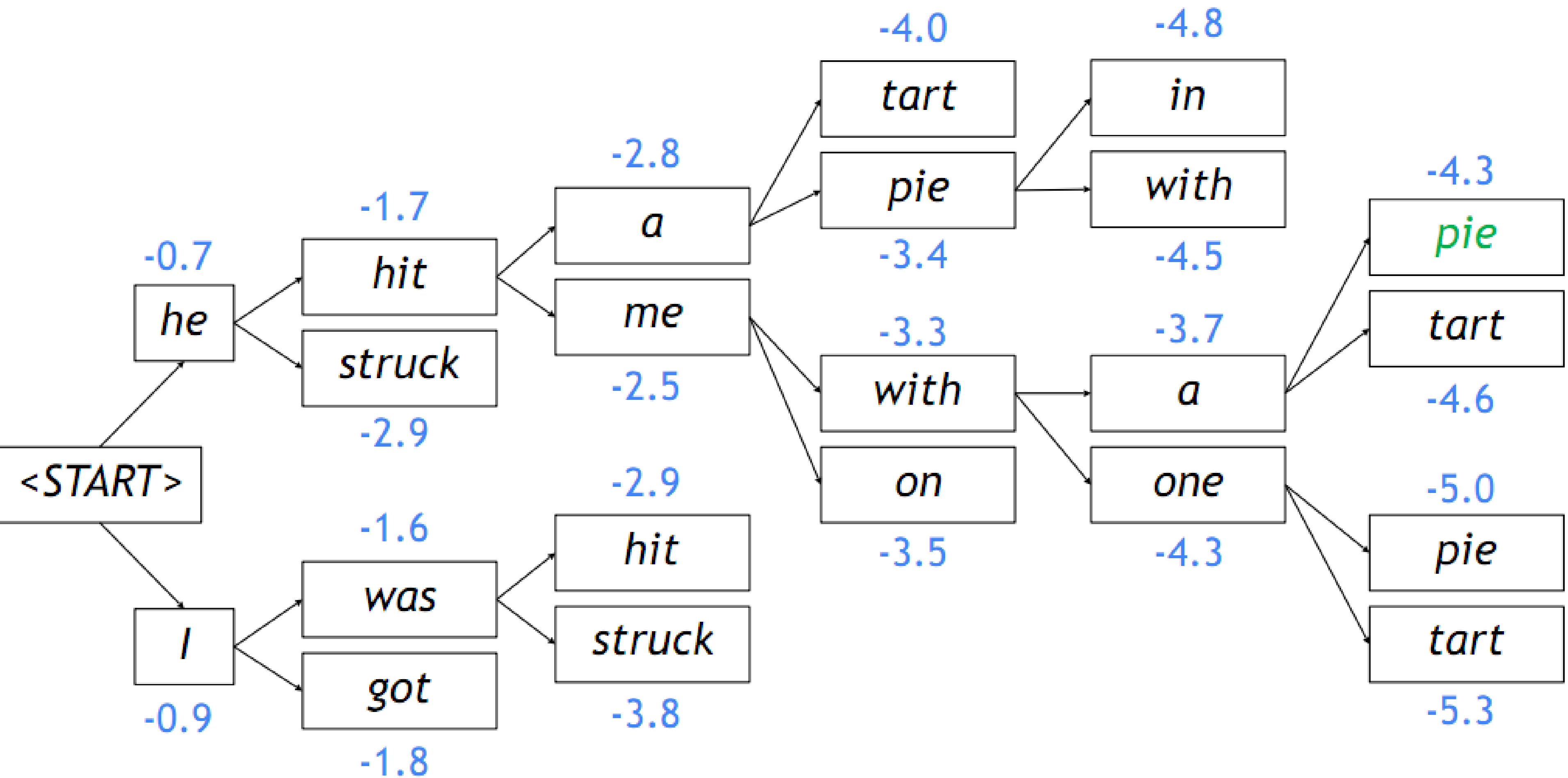
Ví dụ beam search với k = 2

- Với mỗi giả thuyết tìm tiếp *k* giả thuyết tiếp theo có điểm cao nhất



Ví dụ beam search với k = 2

- Giả thuyết có điểm cao nhất là lời giải cần tìm!



Điều kiện dừng beam search

- Trong giải mã tham lam, thường dừng khi mô hình sinh ra token <END>
- Ví dụ: <START> he hit me with a pie <END>
- Đối với beam search, các giả thuyết khác nhau có thể sinh ra token <END> tại các thời điểm khác nhau
- Khi một giả thuyết sinh ra <END> ta gọi giả thuyết đó được hoàn thành và đặt nó sang một bên để tiếp tục tìm các giả thuyết khác
- Thường sẽ dừng beam search khi:
  - Hoặc là đạt đến bước T cho trước
  - Hoặc khi đã tìm ra ít nhất n giả thuyết hoàn thành

Kết thúc beam search

- Khi tìm xong một tập các giả thuyết hoàn thành thì chọn giả thuyết nào?
- Vấn đề: giả thuyết càng dài điểm càng thấp

$$\mathrm{score}(y_1, \ldots, y_t) = \log P_{\mathrm{LM}}(y_1, \ldots, y_t | x) = \sum_{i=1}^{t} \log P_{\mathrm{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$$

- Phương án giải quyết: Chuẩn hóa điểm theo chiều dài giả thuyết

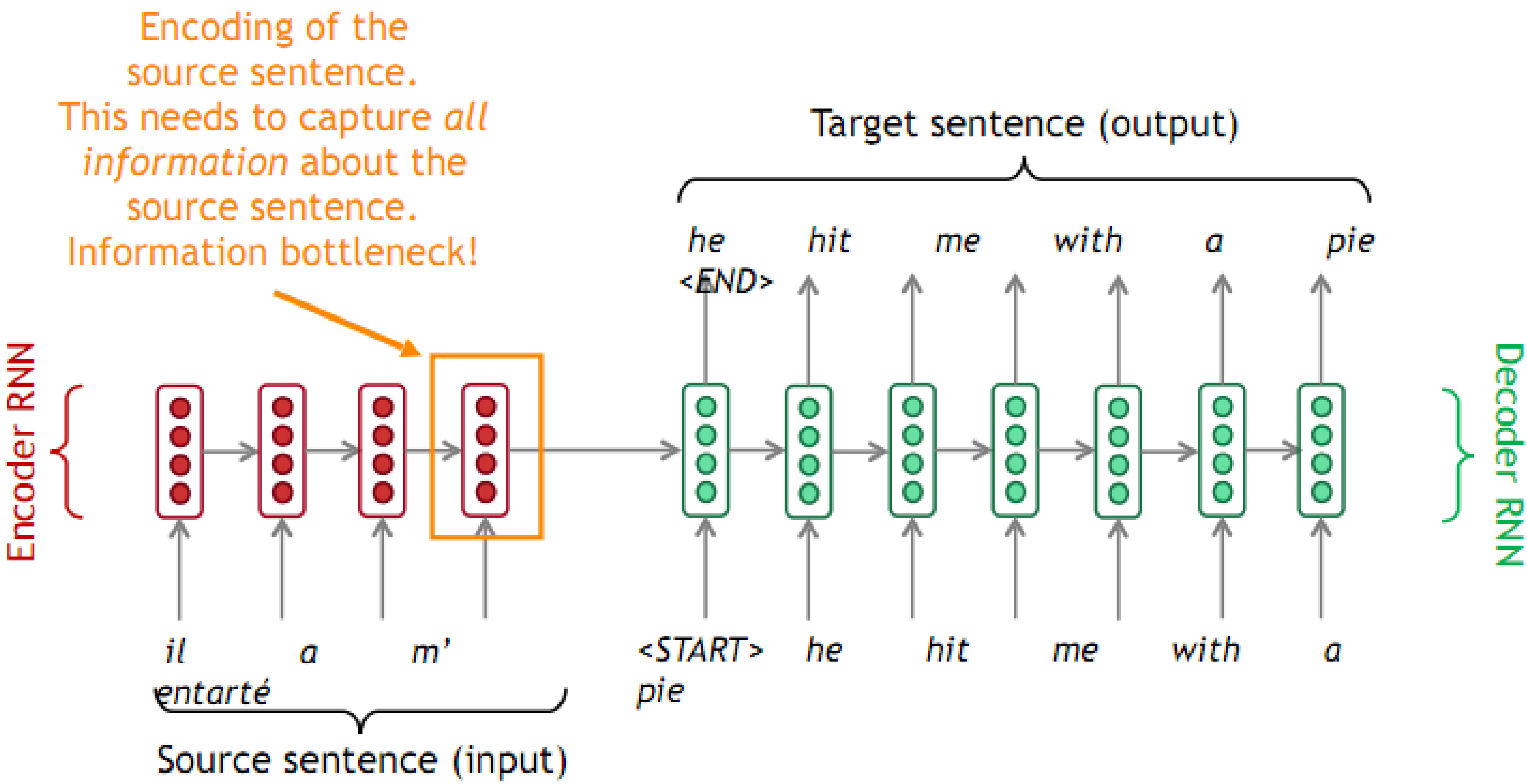
$$\frac{1}{t} \sum_{i=1}^{t} \log P_{\mathrm{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$$

So sánh NMT và SMT

- **Ưu điểm** NMT so với SMT:
    - Hiệu năng tốt hơn: dịch trôi chảy hơn, dùng ngữ cảnh tốt hơn…
    - Chỉ dùng một mạng duy nhất nên có thể huấn luyện end-to-end, không cần tối ưu các mô-đun độc lập nào khác
    - Cần ít sức người hơn: không cần trích xuất đặc trưng thủ công, cùng một phương pháp có thể tái sử dụng cho nhiều cặp ngôn ngữ khác nhau
- **Nhược điểm** NMT so với SMT:
    - NMT khó giải thích hơn, khó gỡ rối
    - NMT khó kiểm soát. Ví dụ: muốn đưa một quy tắc hay gợi ý dịch cho NMT là không dễ dàng.

Cơ chế chú ý
(Attention mechanism)

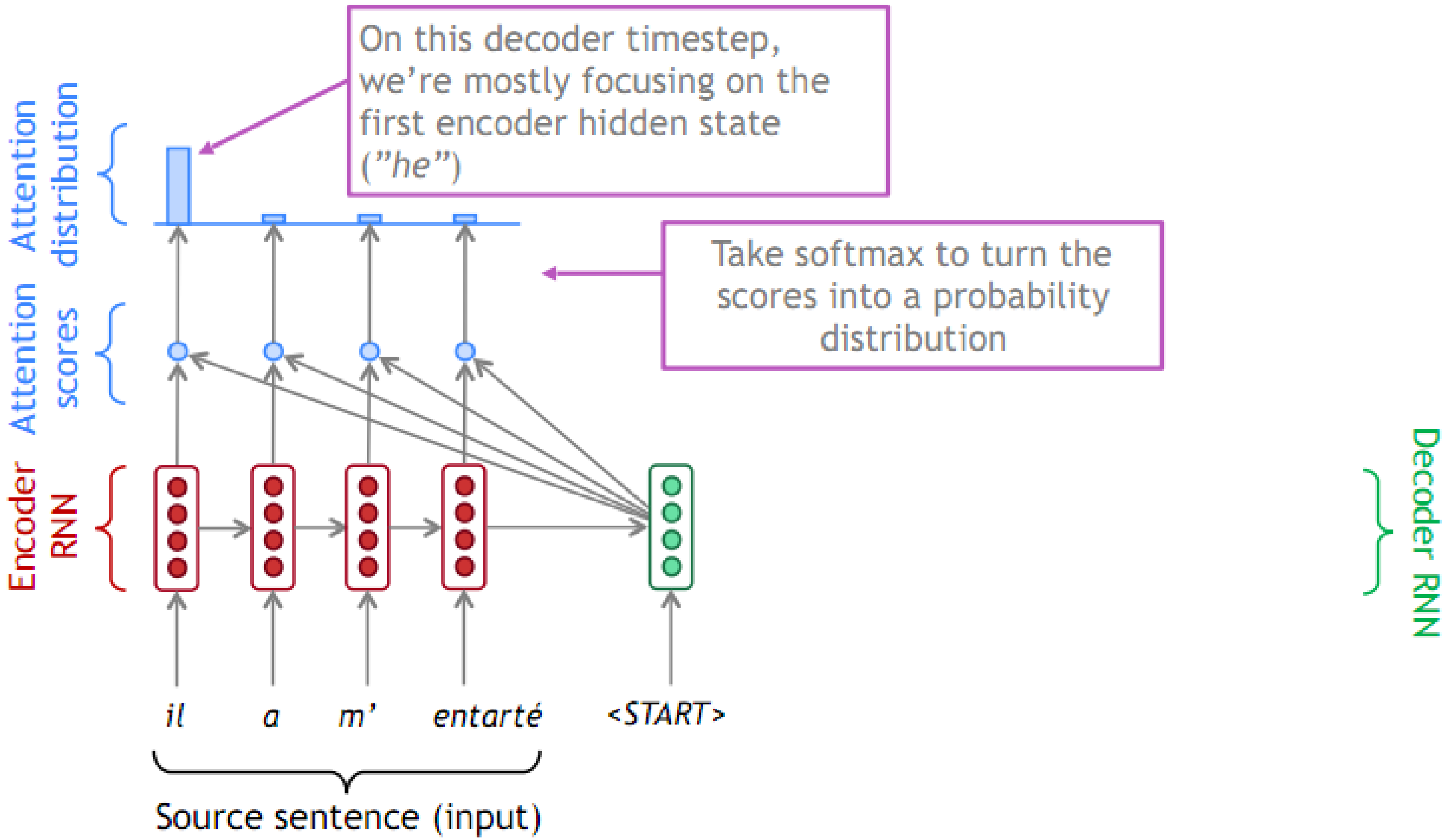
Nút thắt cổ chai của mô hình seq2seq



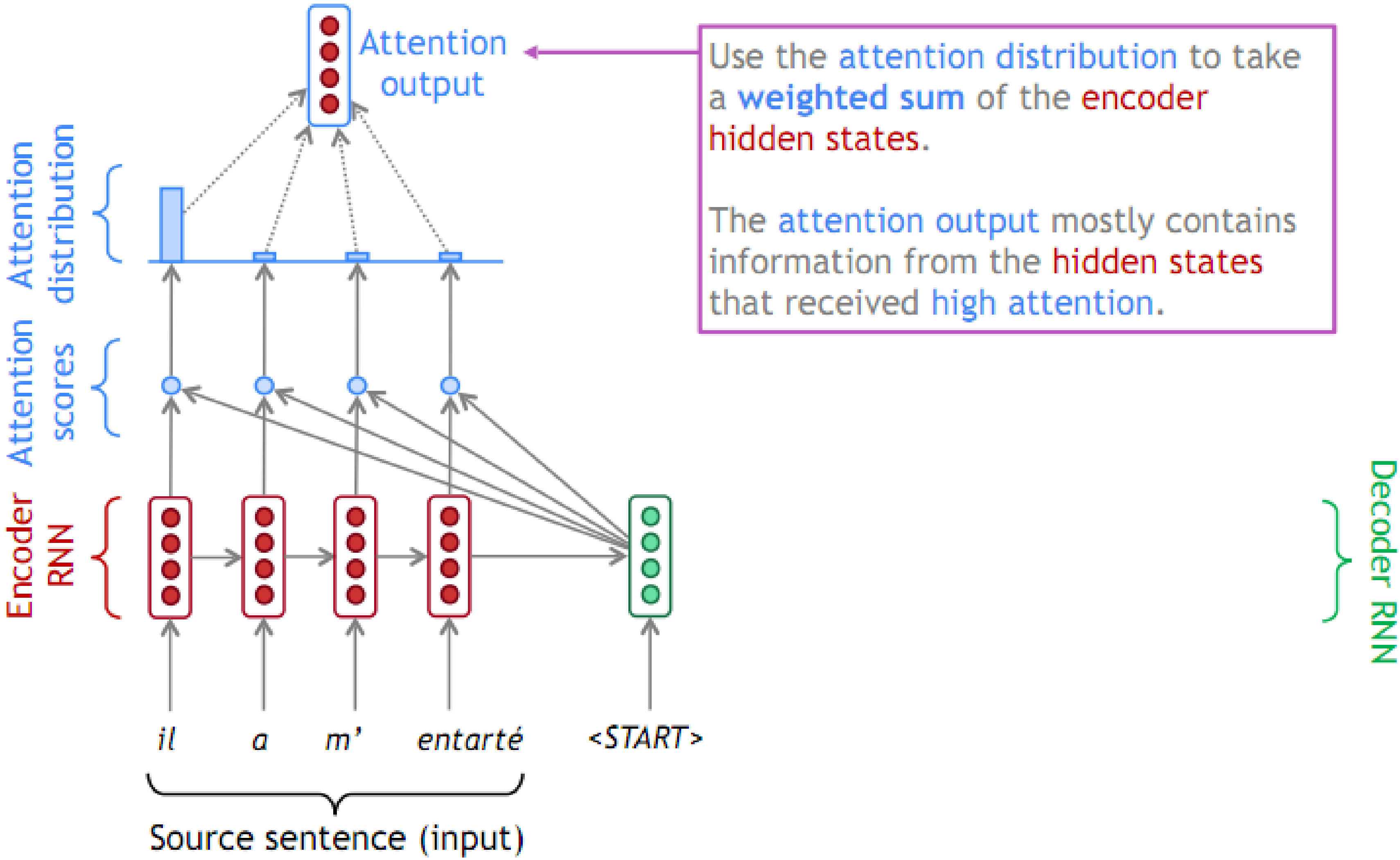
Attention

- **Attention** giải quyết vấn đề nút thắt cổ chai của seq2seq
- **Ý tưởng:** ở mỗi bước giải mã, sử dụng kết nối trực tiếp tới phần mạng mã hóa để tính toán và từ đó chỉ tập trung vào một phần cụ thể câu nguồn, bỏ qua những phần không liên quan.

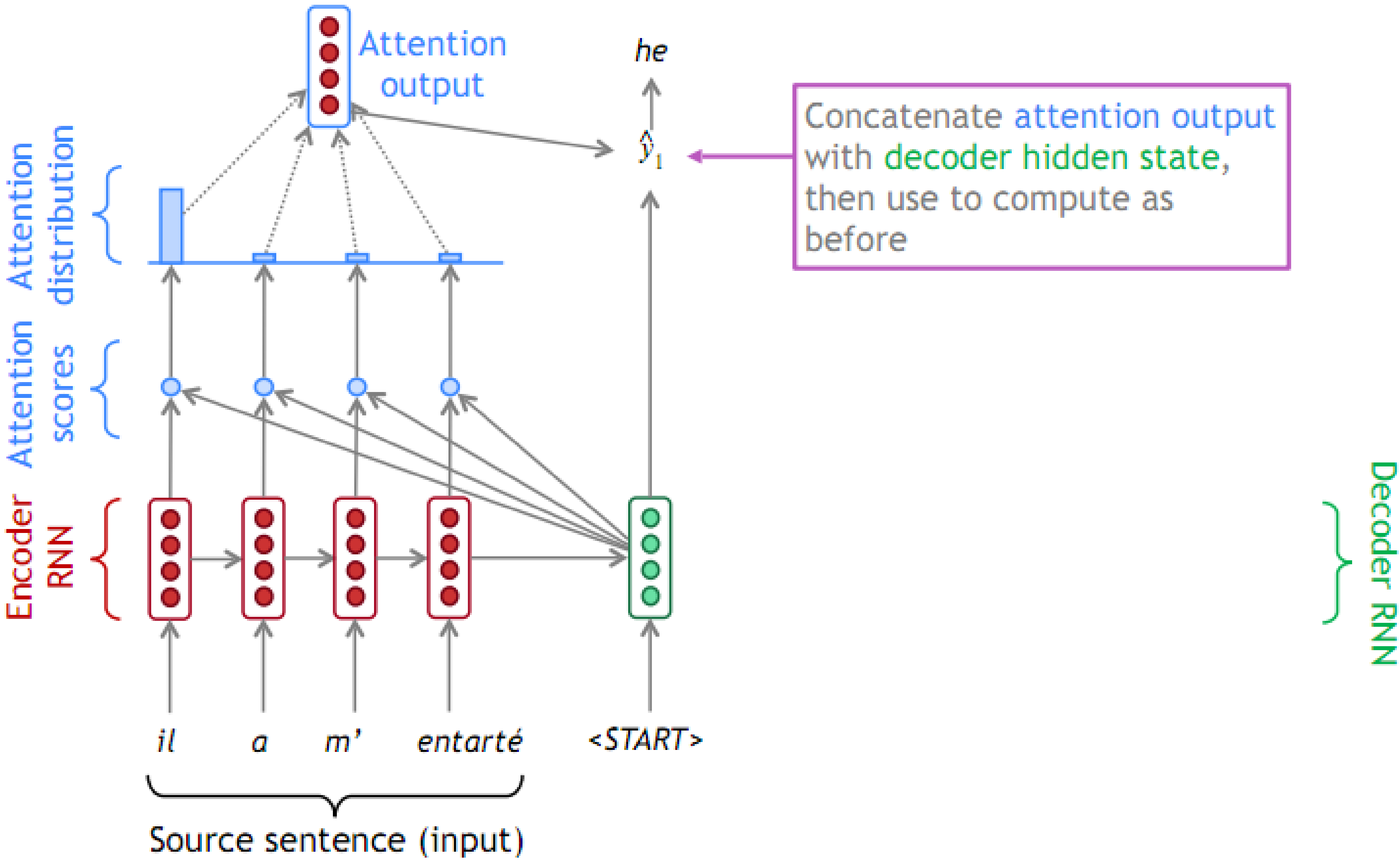
Attention seq2seq



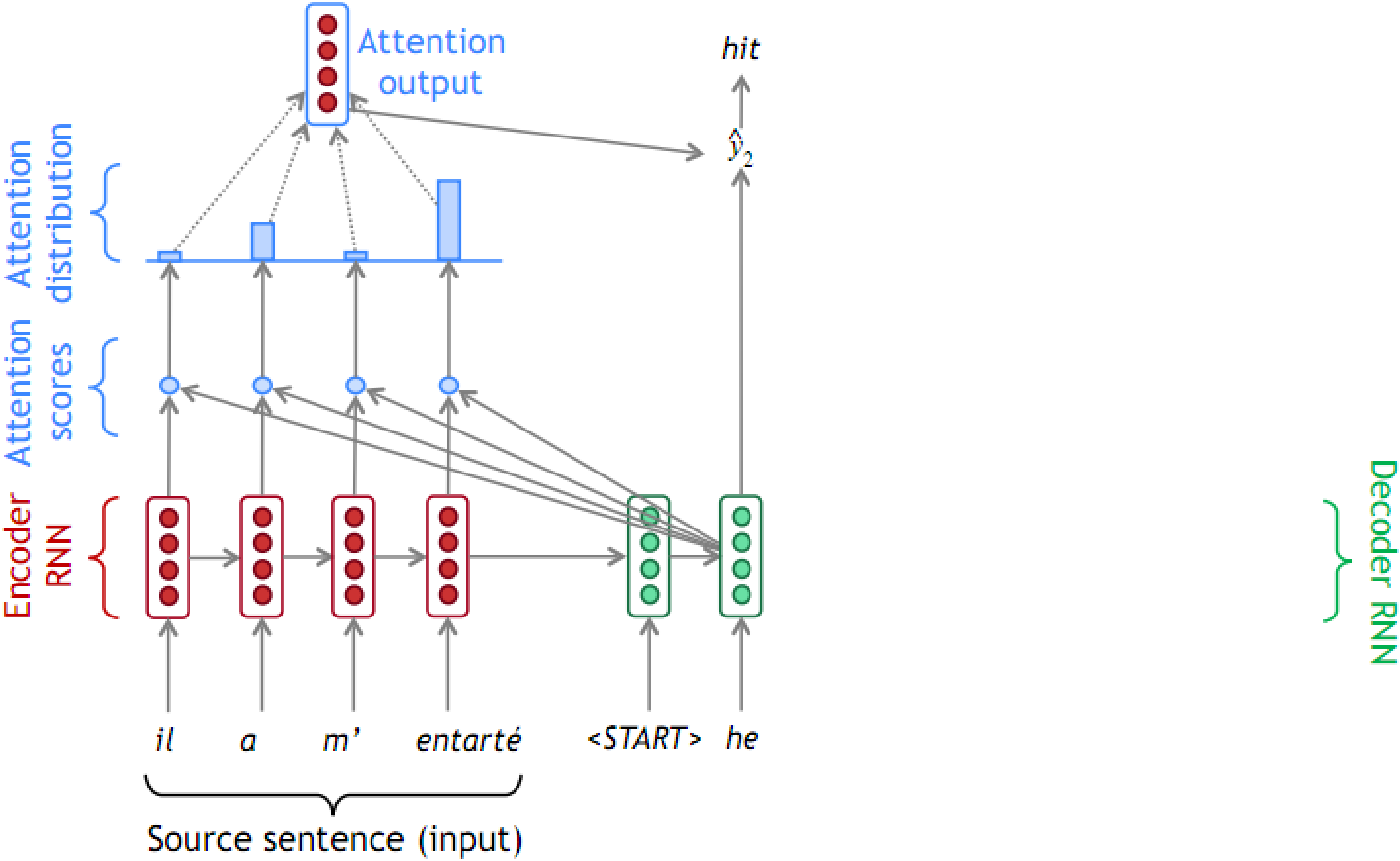
Attention seq2seq



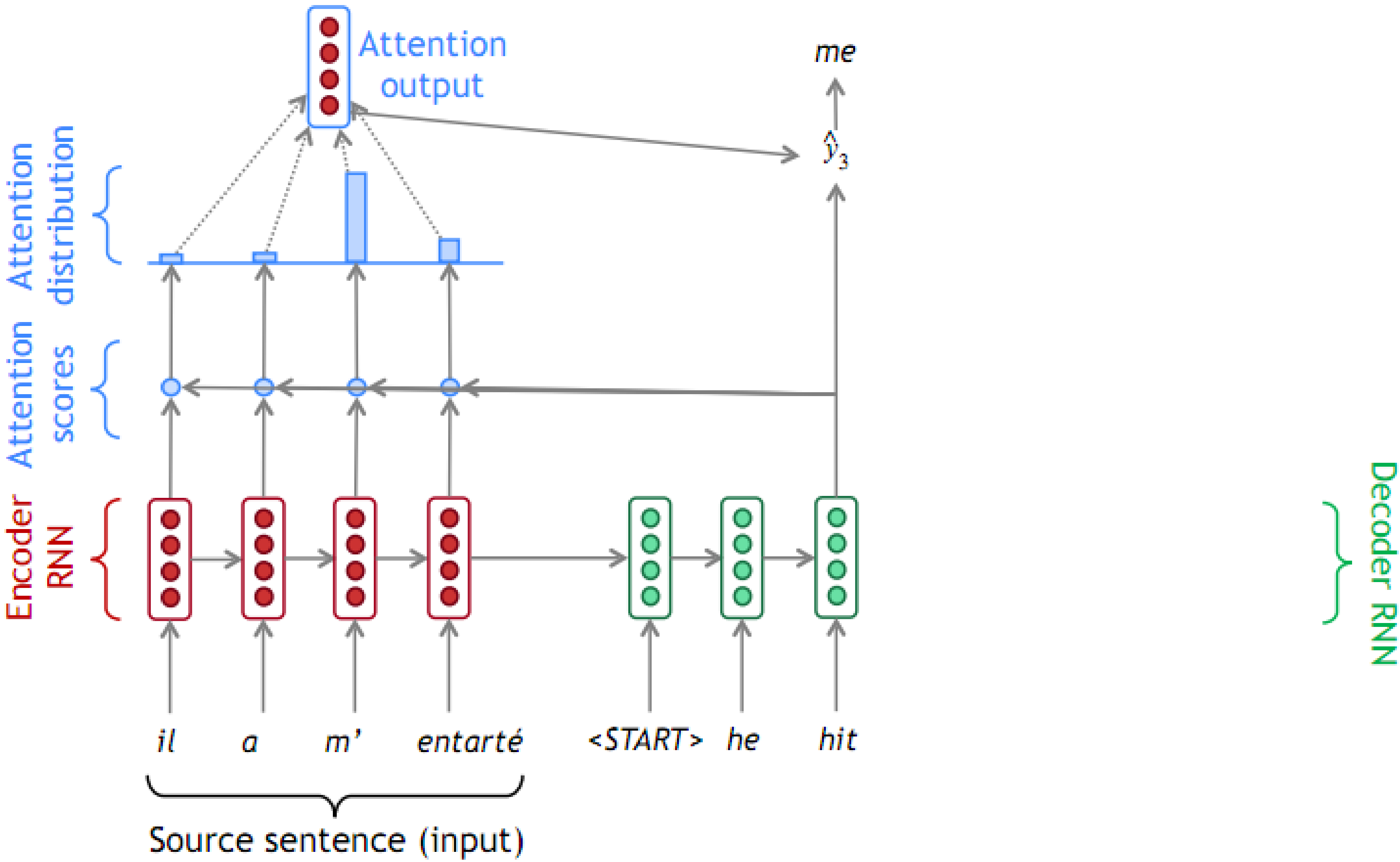
Attention seq2seq



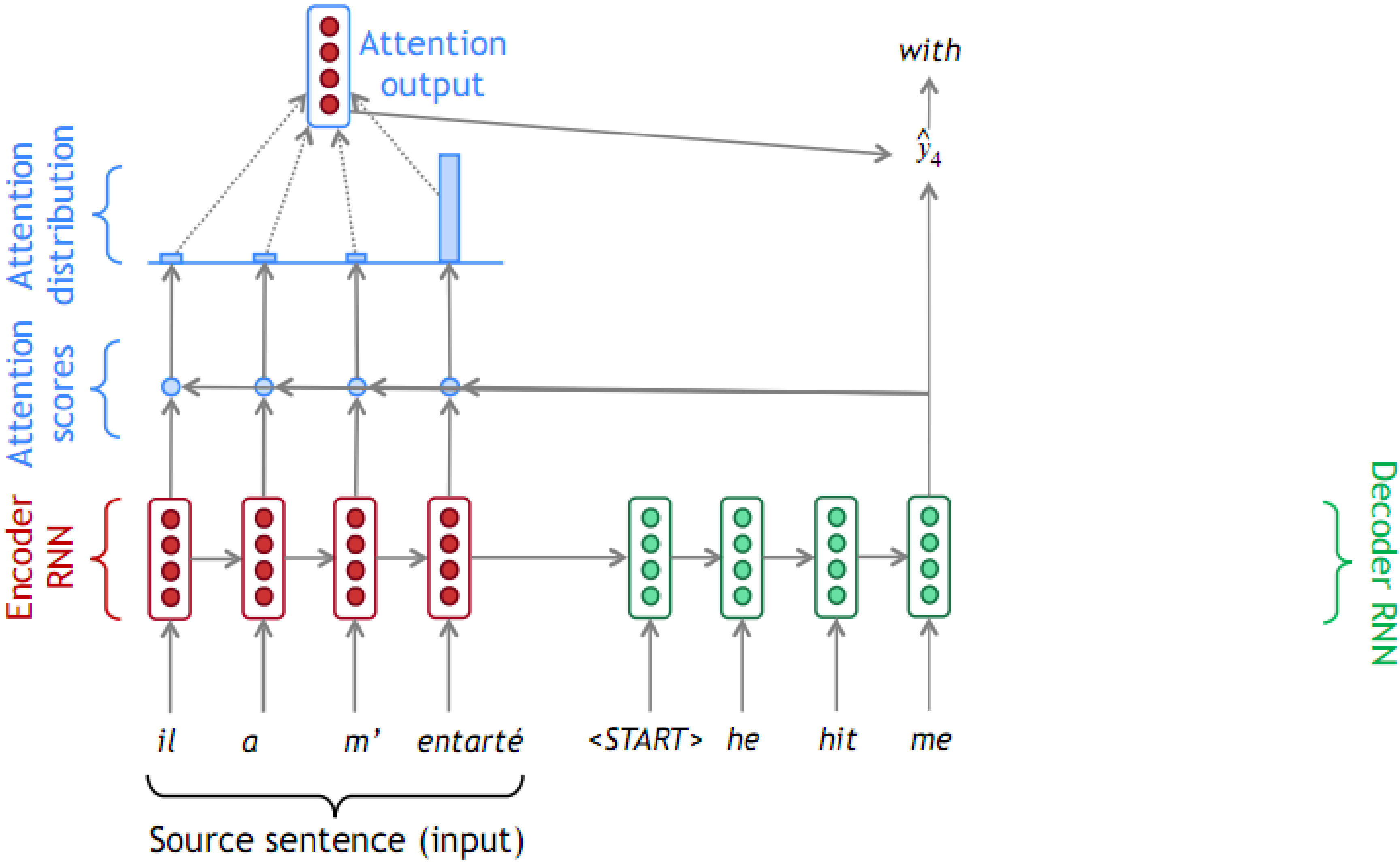
Attention seq2seq



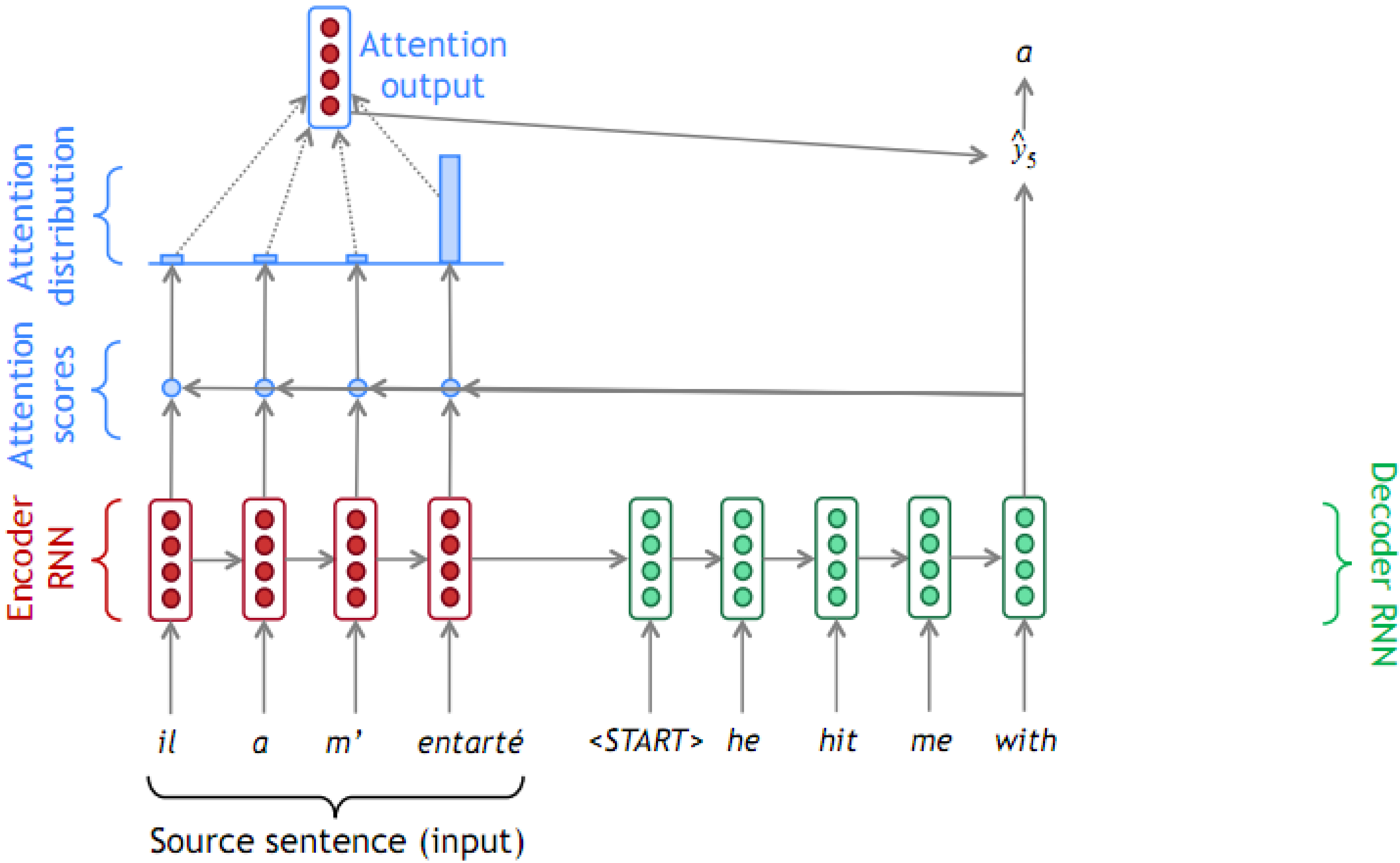
Attention seq2seq



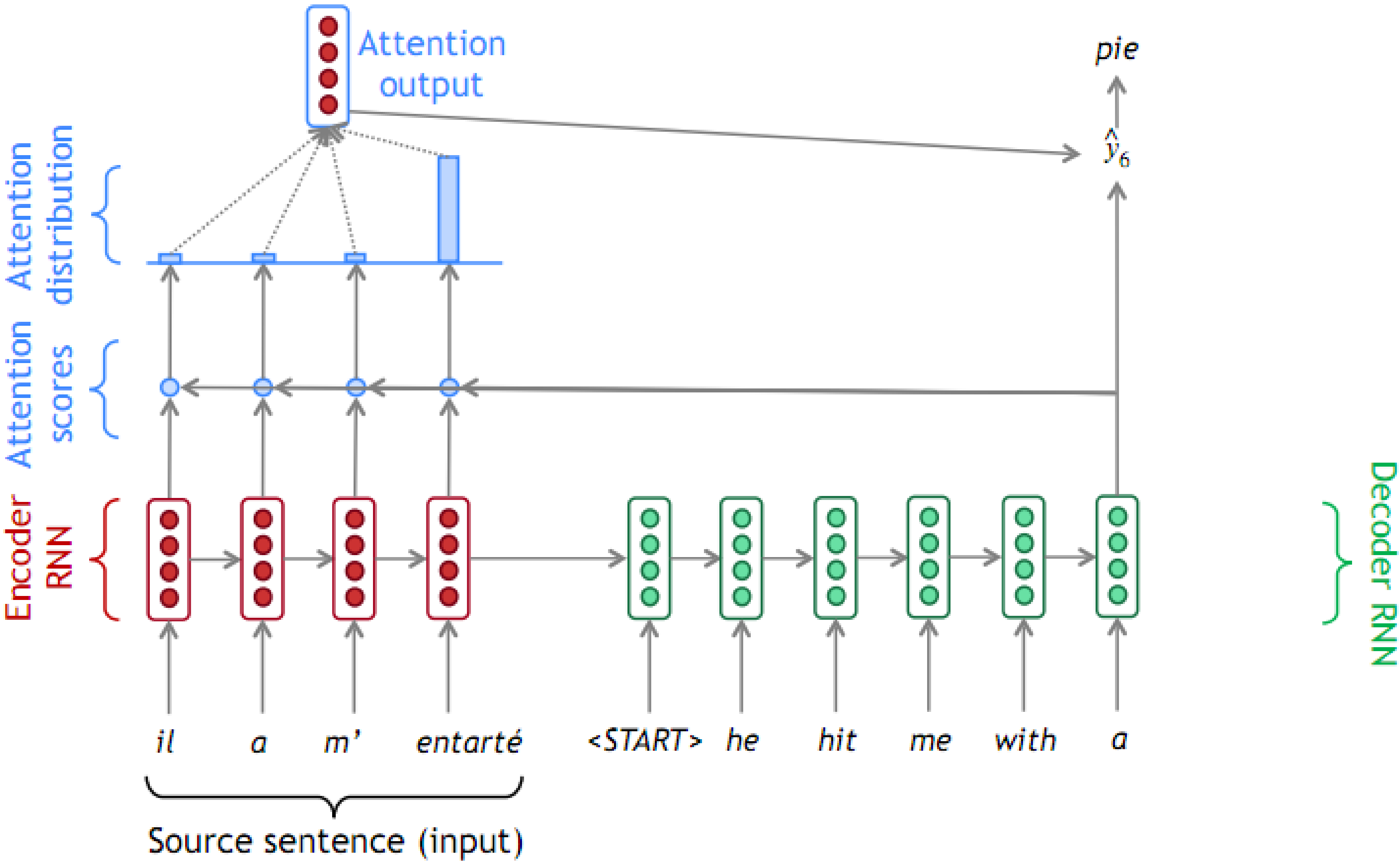
Attention seq2seq



Attention seq2seq



Attention seq2seq



Công thức chi tiết

- We have encoder hidden states $h_1, \ldots, h_N \in \mathbb{R}^h$
- On timestep *t*, we have decoder hidden state $s_t \in \mathbb{R}^h$
- We get the attention scores $\boldsymbol{e}^t$ for this step:

$$\boldsymbol{e}^t = [\boldsymbol{s}_t^T \boldsymbol{h}_1, \ldots, \boldsymbol{s}_t^T \boldsymbol{h}_N] \in \mathbb{R}^N$$

- We take softmax to get the attention distribution α^t for this step (this is a probability distribution and sums to 1)

$$\alpha^t = \mathrm{softmax}(\boldsymbol{e}^t) \in \mathbb{R}^N$$

- We use α^t to take a weighted sum of the encoder hidden states to get the attention output

$$\boldsymbol{a}_t = \sum_{i=1}^{N} \alpha_i^t \boldsymbol{h}_i \in \mathbb{R}^h$$

- Finally we concatenate the attention output $\boldsymbol{a}_t$ with the decoder hidden state s_t and proceed as in the non-attention seq2seq model

$$[\boldsymbol{a}_t; \boldsymbol{s}_t] \in \mathbb{R}^{2h}$$

Tài liệu tham khảo

1. Khóa cs244n của Stanford:

https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1194/

2. Khóa cs231n của Stanford:

http://cs231n.stanford.edu/slides/2020/lecture_10.pdf