hóa (encoder) để chuyển đổi một đối tượng nguồn thành một biểu diễn mã hóa x. Mỗi lần các chuỗi đầu vào được mã hoá, mô hình seq2seq tạo ra một chuỗi mục tiêu sử dụng một bộ giải mã (decoder). Bộ giải mã được giao nhiệm vụ tạo ra một chuỗi từ mục tiêu từ bộ từ vựng mục tiêu. Các từ được tạo ra tuần tự bằng cách điều chỉnh trên biểu diễn đầu vào và trên các từ được tạo ra trước đó. Sau đây là cụ thể về mô hình Sequence-to-Sequence:

Đặt $w_{1:T}$ là chuỗi từ bất kì có độ dài T, $y_{1:T}$ là chuỗi từ mục tiêu cho mỗi input x. Đặt $m_1, \dots, m_T$ là dãy gồm T vector và h_0 là trạng thái khởi tạo của vector. Áp dụng một RNN encoder cho chuỗi như vậy mang lại các trạng thái ẩn h_t tại mỗi bước thời gian t, như sau:

$$h_t \leftarrow RNN(m_t, h_t, \theta)$$

θ là danh sách các tham số của mô hình, được chia theo thời gian. Vector m_t tương ứng với sự gắn kết của một chuỗi từ mục tiêu $w_{1:T}$. Vì vậy, ta có thể viết:

$$h_t \leftarrow RNN(w_t, h_t, \theta)$$

RNN decoder thường được huấn luyện để hoạt động như các mô hình ngôn ngữ có điều kiện. Đó là cố gắng tính toán xác suất của từ mục tiêu thứ *t* đảm bảo điều kiện *x* khi biết *t-1* từ trước đó:

$$p(w_t|w_{1:t-1}, x) = g(w_t, h_{t-1}, x)$$

với một vài tham số của hàm g được tính toán với một lớp affine theo một hàm softmax. Trong việc tính toán xác suất này, trạng thái h_{t-1} đại diện cho các từ đã được huấn luyện trước từ mục tiêu và h_0 được thiết lập thành một số hàm của biến x. Mô hình hoàn chỉnh (bao gồm encoder) được huấn luyện, tương tự như mô hình ngôn ngữ nơ ron, cần phải tối thiểu hóa hàm lỗi (cross-entropy loss) tại mỗi lần lặp trong khi việc điều chỉnh những từ mục tiêu đã được tạo ra trước đó trong tập dữ liệu huấn luyện. Đó là, mô hình được huấn luyện để tối thiểu hóa:

$$-ln \prod_{t=1}^{T} p(y_t|y_{1:t-1}, x)$$

Encoder và decoder khác trọng số với nhau. Trong mô hình seq2seq cơ bản, mỗi đầu vào phải được mã hóa thành một vector trạng thái có chiều dài cố định, bởi vì đó là thứ duy nhất được truyền đến bộ giải mã. Cơ chế Attention cho phép bộ giải mã truy cập trực tiếp vào đầu vào.

***Attention**

Mô hình Sequence-to-Sequence có thể bị phá vỡ khi chuỗi đầu vào quá dài. Nguyên nhân là ở mỗi bước nếu chỉ có một vector ngữ cảnh *c* giao tiếp giữa encoder và decoder, vector đó sẽ phải mã hoá cho toàn bộ chuỗi đầu vào, dẫn đến nó có thể bị tan biến khi nó xử lý chuỗi từ dài. Kĩ thuật Attention (Bahdanau *et al.,* 2016) cho phép bộ giải mã tập trung vào một phần khác nhau từ đầu ra của encoder bằng tìm ra chuỗi vector ngữ cảnh của mỗi bước encoder $c_{1:n}$ với

$$c_i = \sum_{j=1}^{n} \alpha_{ij} h_j$$

Trong đó, tập trọng số α_{ij} của mỗi trạng thái ẩn h_j là đầu ra của một hàm softmax:

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=0}^{n} \exp(e_{ik})}$$

$$e_{ij} = q(s_{i-1}, h_j)$$

với s_{i-1} là trạng thái ẩn tại bước *i-1* của decoder.

(4) Thuật toán Beam Search

Trong mô hình Sequence-to-Sequence, bộ giải mã được điều khiển bởi một câu đã được mã hoá để tạo ra một câu mới. Tại mỗi bước lặp t, bộ giải mã cần đưa ra quyết định từ nào để sinh ra như từ thứ t trong câu. Vấn đề là chúng ta không biết chính xác chuỗi từ cần phải sinh ra để cực đại hoá xác suất có điều kiện tổng thể. Để giải quyết vấn đề này, thuật toán Beam Search Optimization đã được Sam Wiseman và Alexander M. Rush đề xuất (2016). Beam Search với độ rộng K sao cho tại mỗi bước đưa ra K đề xuất và tiếp tục giải mã với một trong số chúng.

Thay vì tính xác suất của từ kế tiếp, chúng ta sử dụng cách tạo ra điểm số (không phải xác suất) cho các chuỗi câu. Mặc định điểm số của một chuỗi bao gồm các từ đứng trước $w_{1:t-1}$ theo sau bởi một từ w_t là $f(w_t, h_{t-1}, x)$, với f là một hàm kiểm tra trạng thái ẩn tại thời điểm $t-1$ của một RNN. Ở đây, hàm f không phải là một hàm softmax để tránh các vấn đề sai lệch nhãn. Thuật toán Beam Search sử dụng trong mô hình Sequence-to-Sequence cụ thể gồm 3 bước chính:

a. Search-Based Loss: sử dụng một hàm tính điểm để tính tổng điểm của mỗi chuỗi.

Cho tập S_t gồm K chuỗi ứng viên có độ dài t. Chúng ta có thể tính được thứ hạng cho mỗi chuỗi trong S_t sử dụng một hàm tuyến tính f với một RNN. Chúng ta khai báo một chuỗi $\hat{y}_{1:t}^{(K)}$ xếp hạng thứ K trong tập S_t dựa vào hàm f. Nghĩa là, ta có khoảng cách tối đa giữa các thứ hạng: