Báo cáo tuần 1 :

**1.Java**

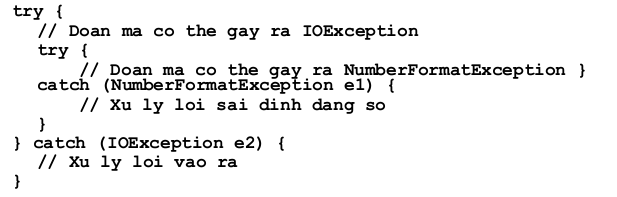
a) Exception Handle:

- Exception

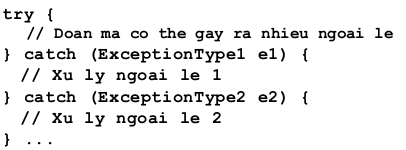
- Xử lí exception :

+ try, catch, finally, throw, throws

+ các khối try-catch lồng nhau : những phần nhỏ trong khối mã sinh ra 1 lỗi, nhưng toàn phần khối mã có thể sinh ra một lỗi khác. Do đó cần có các xử lý ngoại lệ lồng nhau . Khi các khối try-catch lồng nhau, thì khối try bên trong sẽ được thực hiện trước



+nhiều khối catch :một đoạn mã có thể sinh ra hơn 1 ngoại lệ → cần nhiều khối catch



+ khối finally : đảm bảo thực hiện các công việc dù có xảy ra lỗi hay không

- Ủy nhiệm ngoại lệ :

b) Java collection :

- Gồm list ( là 1 tập hợp của các phần tử có thứ tự ), set ( mỗi phần tử trong set chỉ xuất hiện đúng 1 lần ), queure ( kiểu hàng đợi FIFO), map (gồm các cặp key-value , các key không trùng nhau, mỗi key ánh xạ tới 1 value ).

- có hiệu suất (độ phức tạp thời gian) khác nhau

c) Đa luồng

**2. Tf-idf , N-gram**

a) Tf-idf :

- Tf : tần số xuất hiện của 1 từ trong 1 văn bản

- Idf : IDF(t, D) = log( tổng số văn bản trong tập D/ tổng số văn bản xuất hiện từ t)

- Tf-idf = tf\*idf

Tính tf-idf giúp lọc ra những từ khóa của văn bản

b) N-gram:

- cho biết xác suất xuất hiện của 1 câu w1w2...wn là ban nhiêu

P(w1w2…wm) = P(w1) \* P(w2|w1) \* P(w3|w1w2) \*…\* P(wm|w1w2…wm-1)

P(w1w2…wm) = P(w1) \* P(w2|w1) \* P(w3|w1w2) \*…\* P(wm-1|wm-n-1wm-n …wm-2)\* P(wm|wm-nwm-n+1…wm-1) ( dùng xấp xỉ Markov)

-Với công thức này, ta có thể xây dựng mô hình ngôn ngữ dựa trên việc thống kê các cụm có ít hơn n+1 từ. Mô hình ngôn ngữ này gọi là mô hình ngôn ngữ N-gram.

**3. Word2Vec**

Word2vec là một mạng neural 2 lớp với duy nhất 1 tầng ẩn, lấy đầu vào là một corpus lớn và sinh ra không gian vector(với số chiều khoảng vài trăm), với mỗi từ duy nhất trong corpus được gắn với một vector tương ứng trong không gian.

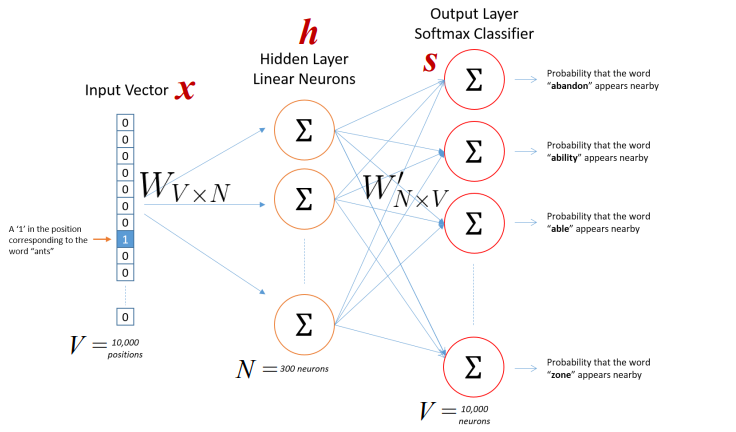
Các word vectors được xác định trong không gian vector sao cho những từ có chung ngữ cảnh trong corpus được đặt gần nhau trong không gian. Dự đoán chính xác cao về ý nghĩa của một từ dựa trên những lần xuất hiện trước đây

Word2vec có 2 model là skip-gram và Cbow:

* Skip-gram model là model predict word surrounding khi cho một từ cho trước
* Cbow là viết tắt của continous bag of word . Model này ngược với model skip-gram tức là cho những từ surrounding predict word current.

Trong thực tế người ta chỉ chọn một trong 2 model để training, Cbow thì training nhanh hơn nhưng độ chính xác không cao bằng skip-gram và ngược lại

**Mô hình toán học của Word2vec**:



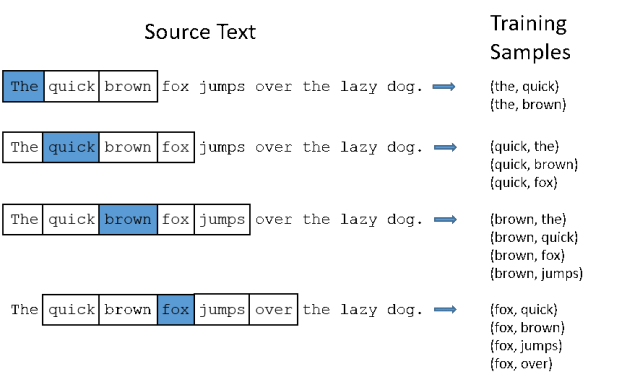
* Input là one-hot-vector mỗi word sẽ có dạng x1, x2,…xv trong đó V là số vocabulary, là một vector trong đó mỗi word sẽ có giá trị 1 tương đương với index trong vocabulary và còn lại sẽ là 0.
* Weight matrix giữa input và hidden layer là matrix W(có dimention VxN) có active function là linear, weight giữa hidden và out put là W′W′ (có dimention là NxV) active function của out put là soft max.

**Nhiệm vụ làm giả**

Đào tạo một neural network để làm như sau. Cho một từ cụ thể ở giữa một câu ( từ đầu vào ), nhìn vào các từ gần đó và chọn một từ ngẫu nhiên. Mạng sẽ cho chúng ta biết xác suất cho mỗi từ trong từ vựng của chúng ta là “từ gần” mà chúng ta đã chọn.

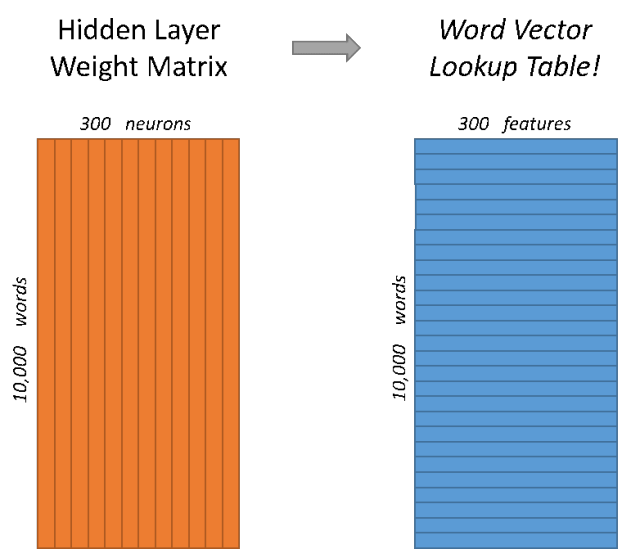
Các xác suất đầu ra sẽ liên quan đến khả năng nó tìm thấy mỗi từ vựng gần từ đầu vào.

Đào tạo mạng thần kinh để thực hiện điều này bằng cách cho các cặp từ đó tìm thấy trong tài liệu đào tạo



Mạng sẽ tìm hiểu số liệu thống kê từ số lần mỗi cặp hiển thị

**Lớp Ẩn**

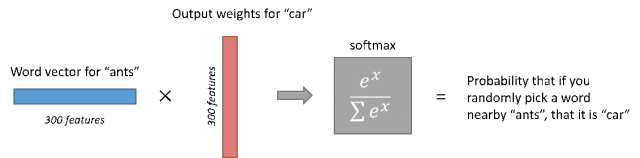


**Lớp đầu ra**

 Lớp đầu ra là một trình phân cụm hồi quy softmax. Có một hướng dẫn sâu về Softmax Regression ở đây, nhưng ý chính của nó là mỗi nơron đầu ra (mỗi từ trong từ vựng của chúng ta!) Sẽ tạo ra một đầu ra giữa 0 và 1, và tổng của tất cả các giá trị đầu ra này sẽ cộng lại được 1.

Mỗi neural đầu ra có một vector trọng số mà nó nhân với vector từ ở lớp ẩn, sau đó nó áp dụng hàm exp(x) cho kết quả. Cuối cùng, để có được kết quả đầu ra tổng là 1, chúng tôi chia kết quả này cho tổng tất cả kết quả 10.000 nút đầu ra.

 Minh họa về tính toán kết quả của neuron đầu ra cho từ “car”



\*) Note : phần này e dịch ra + tham khảo các nguồn nên e mới chỉ hiểu í tưởng + cách thức lớp ẩn hoạt động + tính toán.