Recursive Neural Networks

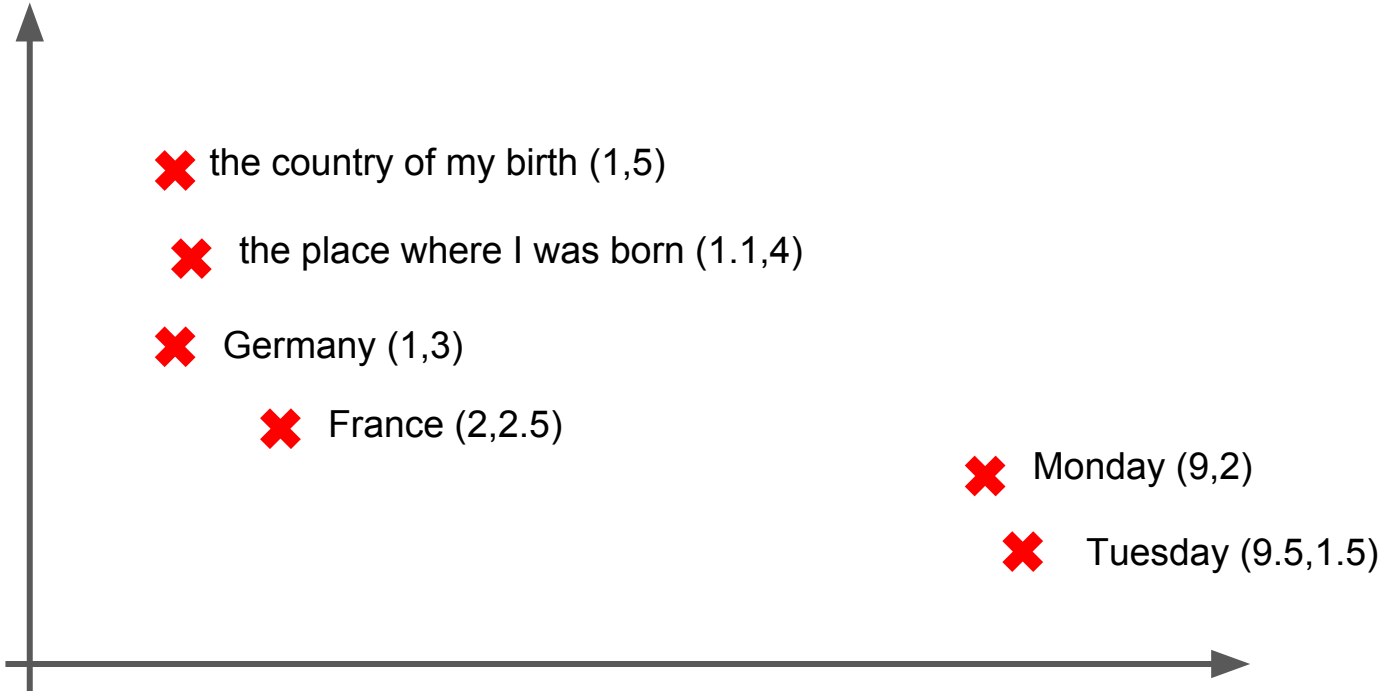
Seminario de Aprendizaje Profundo para PLN

Grupo de PLN, InCo, UdelaR, 2016

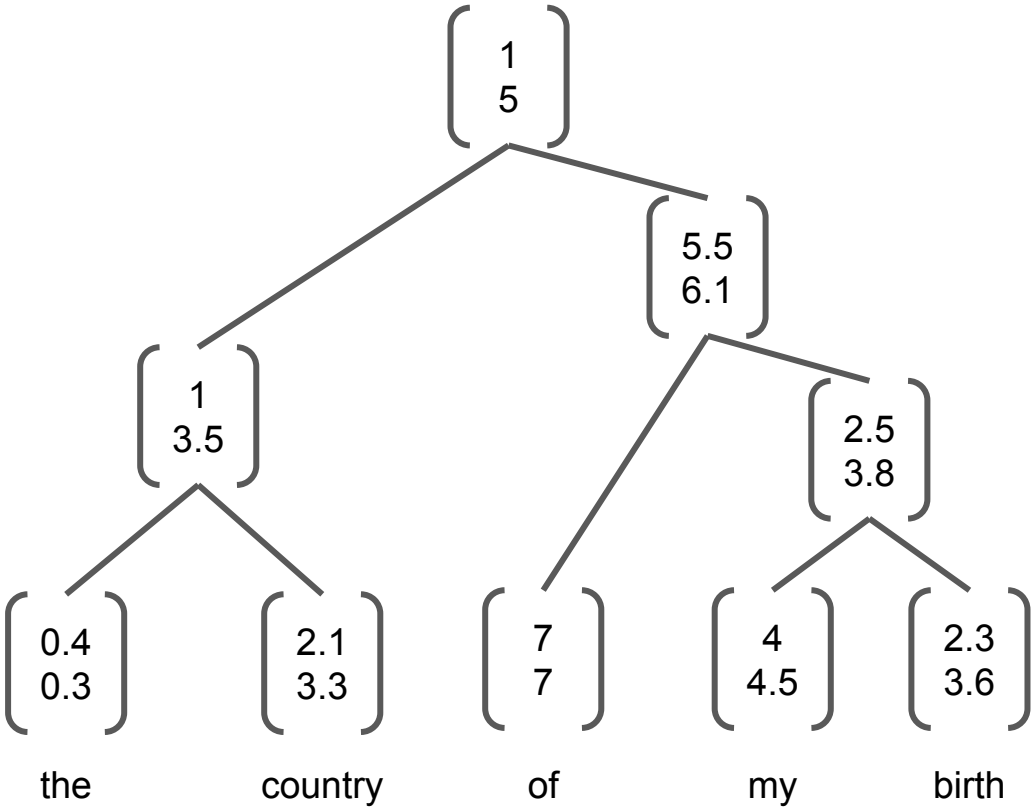
Agenda

- Phrase embeddings
- Recurrent Neural Networks
- Parsing con RNN
- Compositional Vector Grammars
- Análisis de sentimiento con RNN
- Matrix-Vector Recursive Neural Network
- Recurrent Neural Tensor Network

Phrase embeddings



Phrase embeddings



Recursive Neural Network

Comenzamos con un conjunto de word embeddings

$x = (x_1,...x_n) \in R^n$

Y un conjunto de reglas de la forma

$p \rightarrow c_1 c_2$

c_1 y c_2 pueden ser nodos terminales (x) o cualquier nodo intermedio

Recursive Neural Network

A cada nodo intermedio p le va a corresponder un vector

$$p = \tanh(W[c_1;c_2] + b)$$

$p \in R^n$ igual que los x, para poder combinarlos

$[c_1;c_2]$ nota los vectores concatenados, por lo tanto $\in R^{2n}$

$W \in R^{n*2n}$

$b \in R^n$

Recursive Neural Network

¿Qué son estos vectores p?

Embeddings de frases

my birth → (2.5, 3.8)

of my birth → (5.5, 6.1)

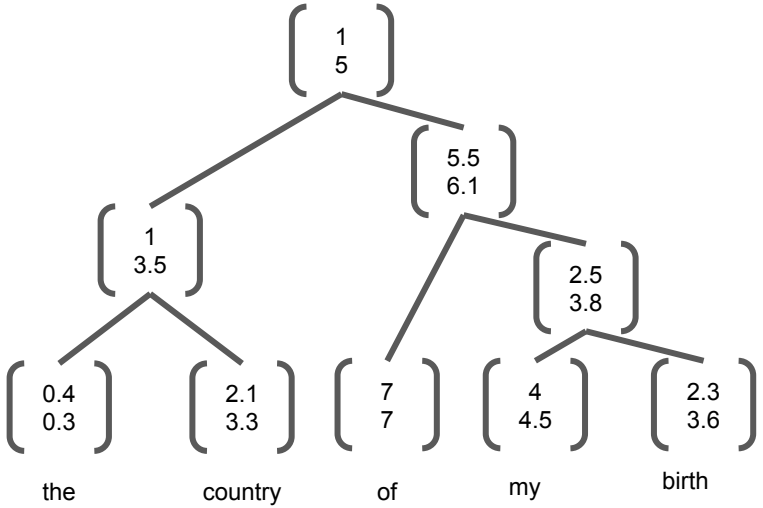
the country → (1, 3.5)

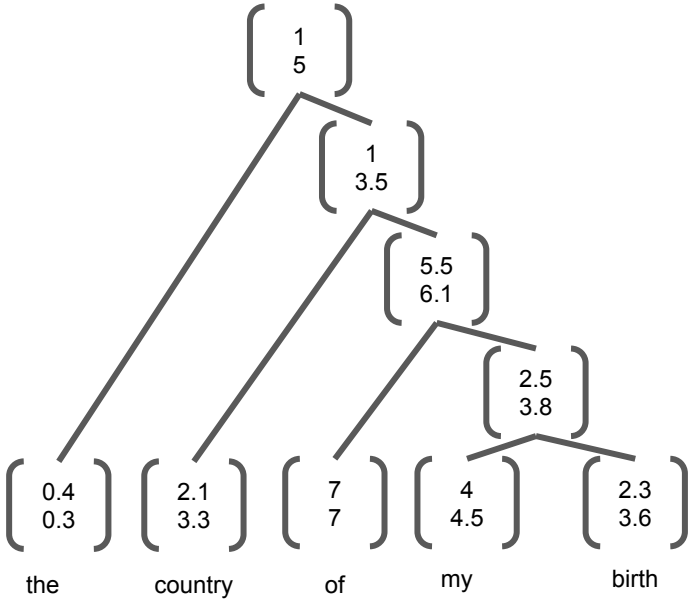
the country of my birth → (1, 5)

Parsing con RNN

Necesitamos aprender los pesos de la matriz W

Pero además necesitamos conocer los árboles para saber en qué orden realizar los cálculos





Parsing con RNN

Socher, Manning, Ng 2010 plantean cuatro modelos

- Model 1: Greedy RNN
- Model 2: Greedy, Context-sensitive RNN
- Model 3: Greedy, Context-sensitive RNN and Category Classifier
- Model 4: Global, Context-sensitive RNN and Category classifier

1 - Greedy RNN

Vamos a aprender la matriz W, y otro vector W^{score}, tal que

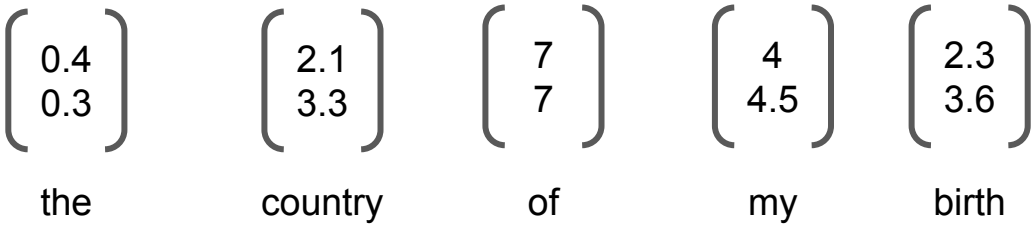
$W^{score} \in R^{1*n}$

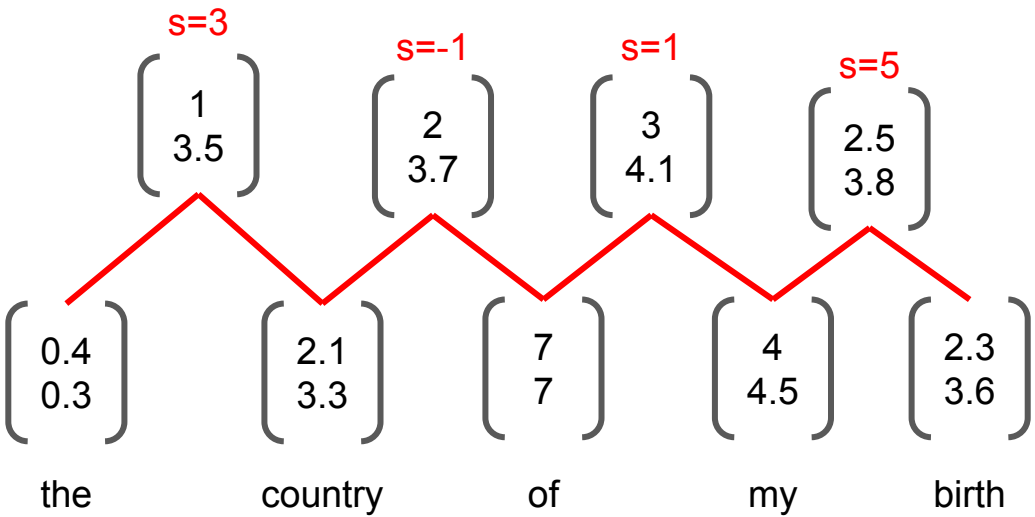
Se usa para calcular

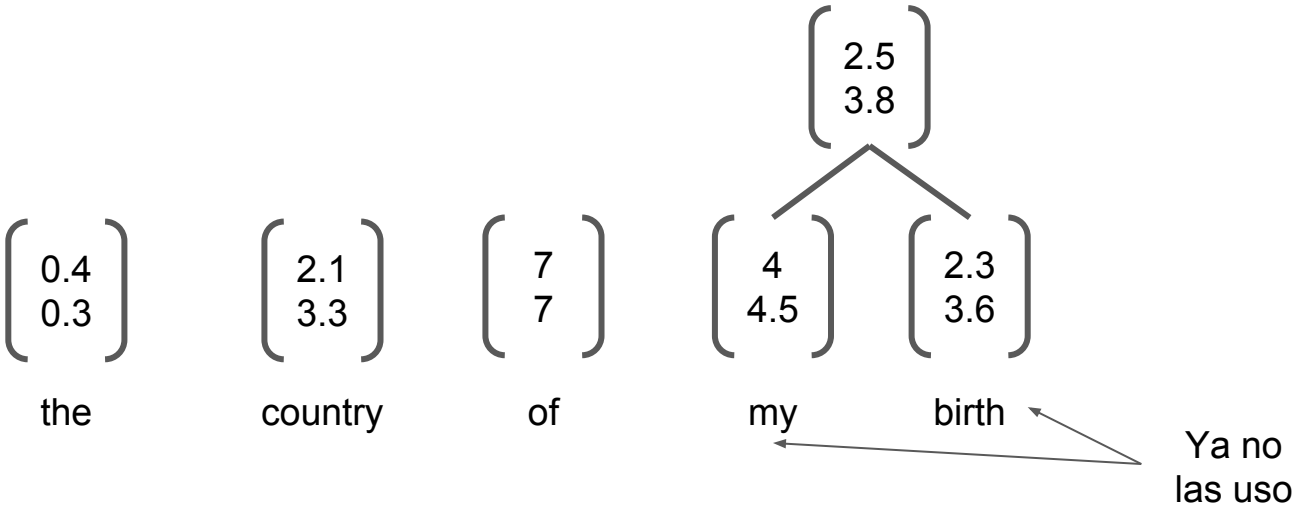
$s_{i,j}=W^{score}*p$

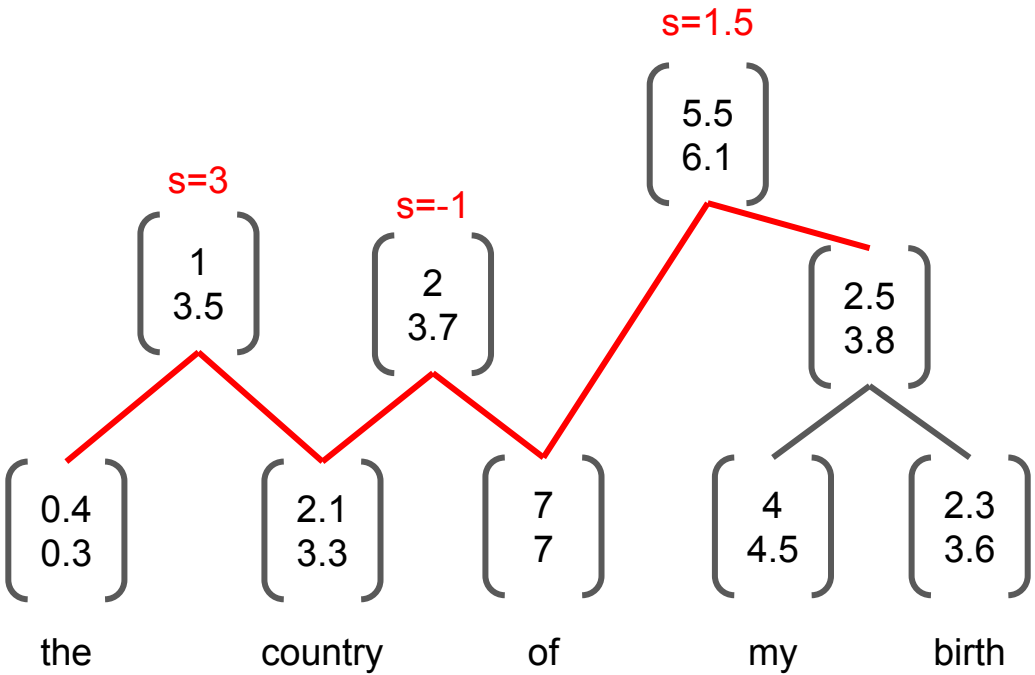
Donde $s_{i,j}$ es el score de todo el subárbol representado por p

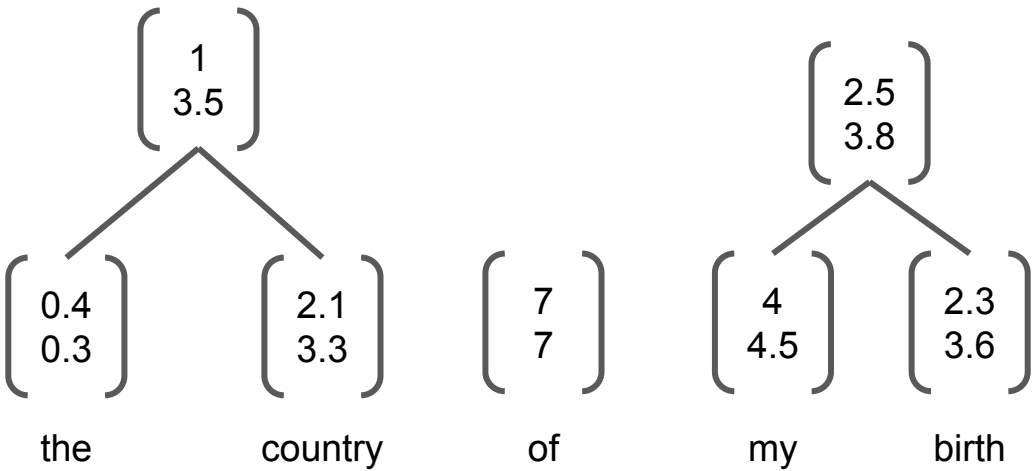
La idea es entrenar para que $s_{i,j}$ sea mayor para frases existentes en los análisis del corpus y menor para frases que no son válidas

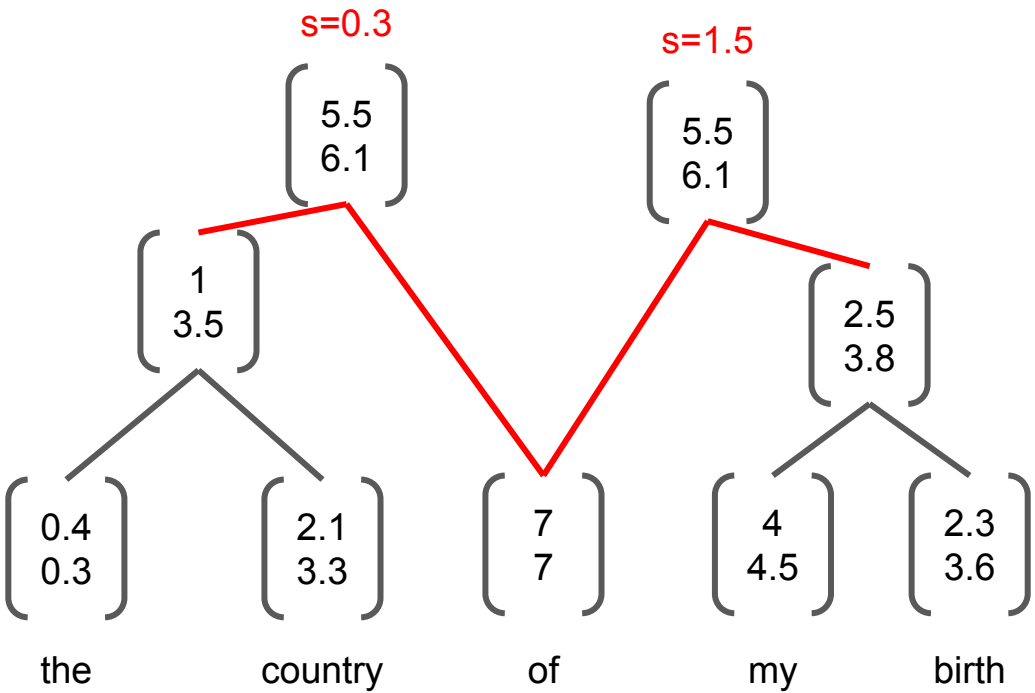

0.4
0.3
2.1
3.3
7
7
4
4.5
2.3
3.6
the
country
of
my
birth

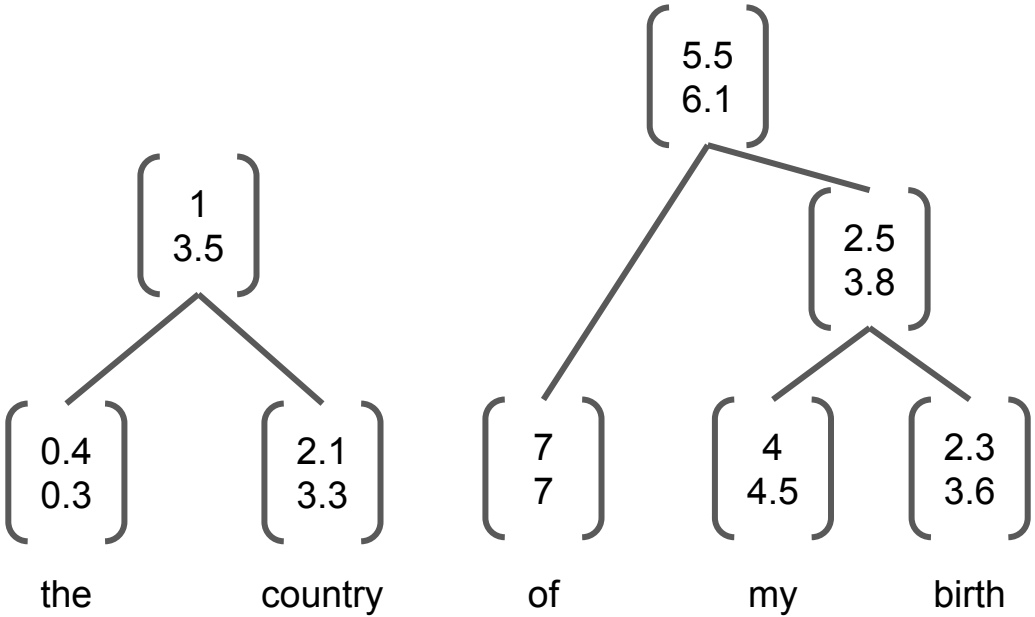

s=3
s=-1
s=1
s=5
1
3.5
2
3.7
3
4.1
2.5
3.8
0.4
0.3
2.1
3.3
7
7
4
4.5
2.3
3.6
the
country
of
my
birth

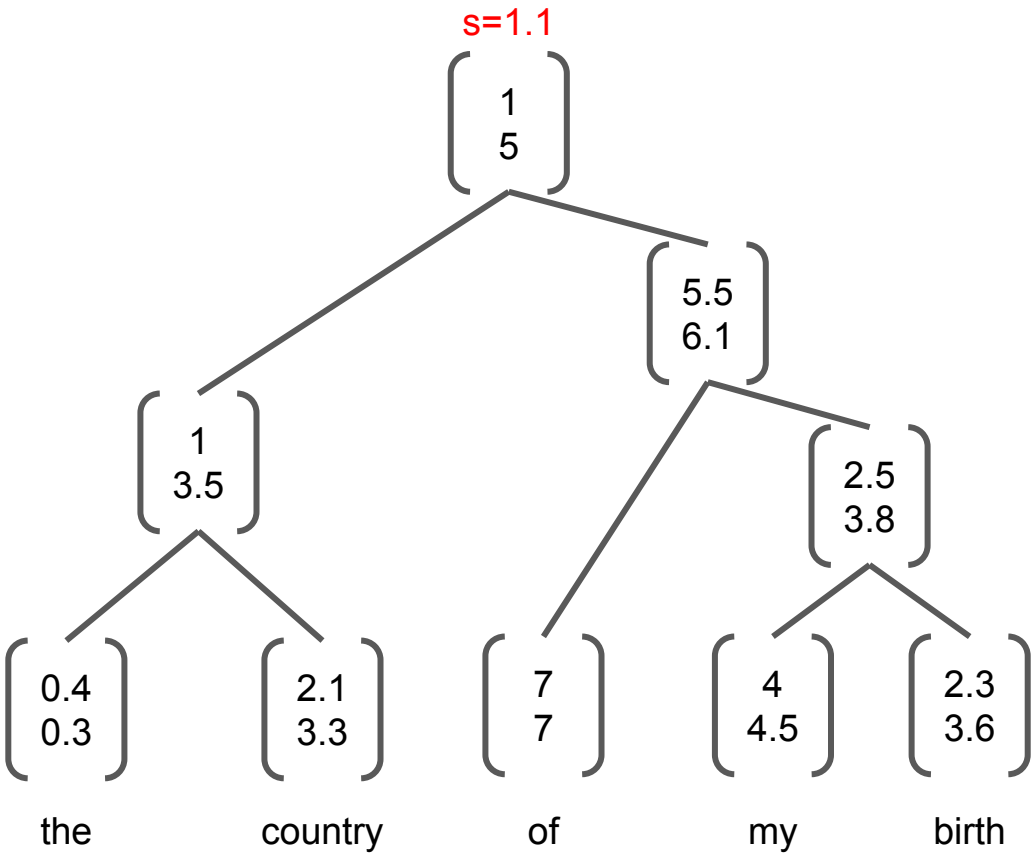

2.5
3.8
0.4
0.3
2.1
3.3
7
7
4
4.5
2.3
3.6
the
country
of
my
birth
Ya no
las uso


s=1.5
5.5
6.1
s=3
1
3.5
s=-1
2
3.7
2.5
3.8
0.4
0.3
2.1
3.3
7
7
4
4.5
2.3
3.6
the
country
of
my
birth


1
3.5
0.4
0.3
2.1
3.3
7
7
2.5
3.8
4
4.5
2.3
3.6
the
country
of
my
birth


s=0.3
s=1.5
5.5
6.1
5.5
6.1
1
3.5
2.5
3.8
0.4
0.3
2.1
3.3
7
7
4
4.5
2.3
3.6
the
country
of
my
birth


5.5
6.1
1
3.5
2.5
3.8
0.4
0.3
2.1
3.3
7
7
4
4.5
2.3
3.6
the
country
of
my
birth


s=1.1
1
5
5.5
6.1
1
3.5
2.5
3.8
0.4
0.3
2.1
3.3
7
7
4
4.5
2.3
3.6
the
country
of
my
birth

1 - Greedy RNN

Cómo lo entrenamos?

Es un misterio…

El paper dice que los detalles de entrenamiento están en la sección 3, pero la sección 3 no habla nunca de este modelo

2 - Greedy, Context-sensitive RNN

Extiende el modelo greedy para agregar algunas palabras más de contexto

Pueden ser varias palabras, pero por ejemplo incluimos la palabra a la izquierda y a la derecha de la frase

$$p = \tanh(W[x_{-1};c_1;c_2;x_{+1}] + b)$$

$[x_{-1};c_1;c_2;x_{+1}]$ ahora concatena cuatro vectores

$W \in R^{n*4n}$

$p \in R^n$, $W^{score} \in R^{1*n}$ (igual que antes)

3 - Greedy, Context-sensitive and category classifier

Se agrega una capa nueva tipo softmax que se aplica a los p

Esta capa clasifica entre las posibles categorías de los constituyentes

NP, VP, PP, ...

Es entrenamiento supervisado, ya que en el corpus están las categorías correctas

Durante el entrenamiento, el aprendizaje en esta capa también influirá en los pesos de las capas de arriba (W)

4 - Global, Context-sensitive and category classifier

En vez de usar el algoritmo greedy, podemos usar programación dinámica para encontrar el mejor árbol dadas las diferentes combinaciones de score

Pero nuestros W y W^{score} están entrenados para la versión greedy

Habrán hecho la prueba usando esos mismos pesos a ver si mejora?

Entrenamiento con *global, regularized risk objective in a max-margin framework*

4 - Global, Context-sensitive and category classifier

El corpus está compuesto de pares (x_i, y_i) donde x_i es una oración de ejemplo e y_i es su árbol sintáctico correcto

$A(x_i)$ es el conjunto de todos los árboles (correctos o no) que se pueden construir a partir de x_i

T(y) es el conjunto de spans no terminales (pares nodo izquierdo - nodo derecho) que existen en el árbol y

$$\Delta(y,y_i) = \sum_{d \in T(y)} \lambda * 1\{d \notin T(y_i)\}$$

O sea, el delta mide la distancia entre un árbol candidato y el correcto

4 - Global, Context-sensitive and category classifier

Definimos el score total de un árbol como la suma de los scores de cada constituyente, o sea cada span del árbol

$$s(x_i,y_i) = \sum_{d \in T(yi)} s_d(c_1,c_2)$$

El objetivo a maximizar durante el entrenamiento combina las dos cosas

$$J = \sum_i s(x_i,y_i) - \max_{y \in A(xi)} s(x_i,y) + \Delta(y,y_i)$$

4 - Global, Context-sensitive and category classifier

Cómo lo entrenamos?

Hay más detalles, pero es igual de misterioso que el otro modelo

$$\partial/\partial W\ J_i = \sum_{d \in T(yi)} [c_1;c_2]^{(d)} [\boldsymbol{\delta}^{(d+1)}]^T_{1:n} + CW$$

$$\boldsymbol{\delta}^{(d)} = (W^T \boldsymbol{\delta}^{(d+1)}) \circ f'([c_1;c_2]^{(d)})$$

C es una constante, $[.]_{1:n}$ denota los primeros n elementos de un vector

Qué representa este $\boldsymbol{\delta}$?

Antes teníamos $\boldsymbol{\delta}$ para los nodos de salida y otro para los intermedios...

4 - Global, Context-sensitive and category classifier

Experimentos

Word embeddings pre-entrenados con English Gigaword (~600M palabras)

- Vocabulario de 16000 palabras
- Representación en 100 dimensiones

Algoritmo entrenado sobre sección WSJ del Penn Tree Bank

- Oraciones solo hasta largo 15

4 - Global, Context-sensitive and category classifier

Resultados

Método	F1
Model 1 - Greedy RNN	76.55
Model 2 - Greedy, Context-sensitive RNN	83.36
Model 3 - Greedy, Context-sensitive RNN + category classifier	87.05
Model 4 - Global, Context-sensitive RNN + category classifier	92.06
Current Implementation of the Stanford Parser [KM03]	**93.98**

4 - Global, Context-sensitive and category classifier

Ejemplos de phrase embeddings

Sales grew almost 2% to 222.2 million from 222.2 million.

- *Sales surged 22% to 222.22 billion yen from 222.22 billion.*
- *Revenue fell 2% to 2.22 billion from 2.22 billion.*
- *Sales rose more than 2% to 22.2 million from 22.2 million.*
- *Volume was 222.2 million shares, more than triple recent levels.*

4 - Global, Context-sensitive and category classifier

Ejemplos de phrase embeddings

Hess declined to comment.

- *PaineWebber declined to comment.*
- *Phoenix declined to comment.*
- *Campeau declined to comment.*
- *Coastal would n’t disclose the terms.*

4 - Global, Context-sensitive and category classifier

Ejemplos de phrase embeddings

We were lucky

- *It was chaotic*
- *We were wrong*
- *People had died*
- *They still are*

Syntactically Untied RNN

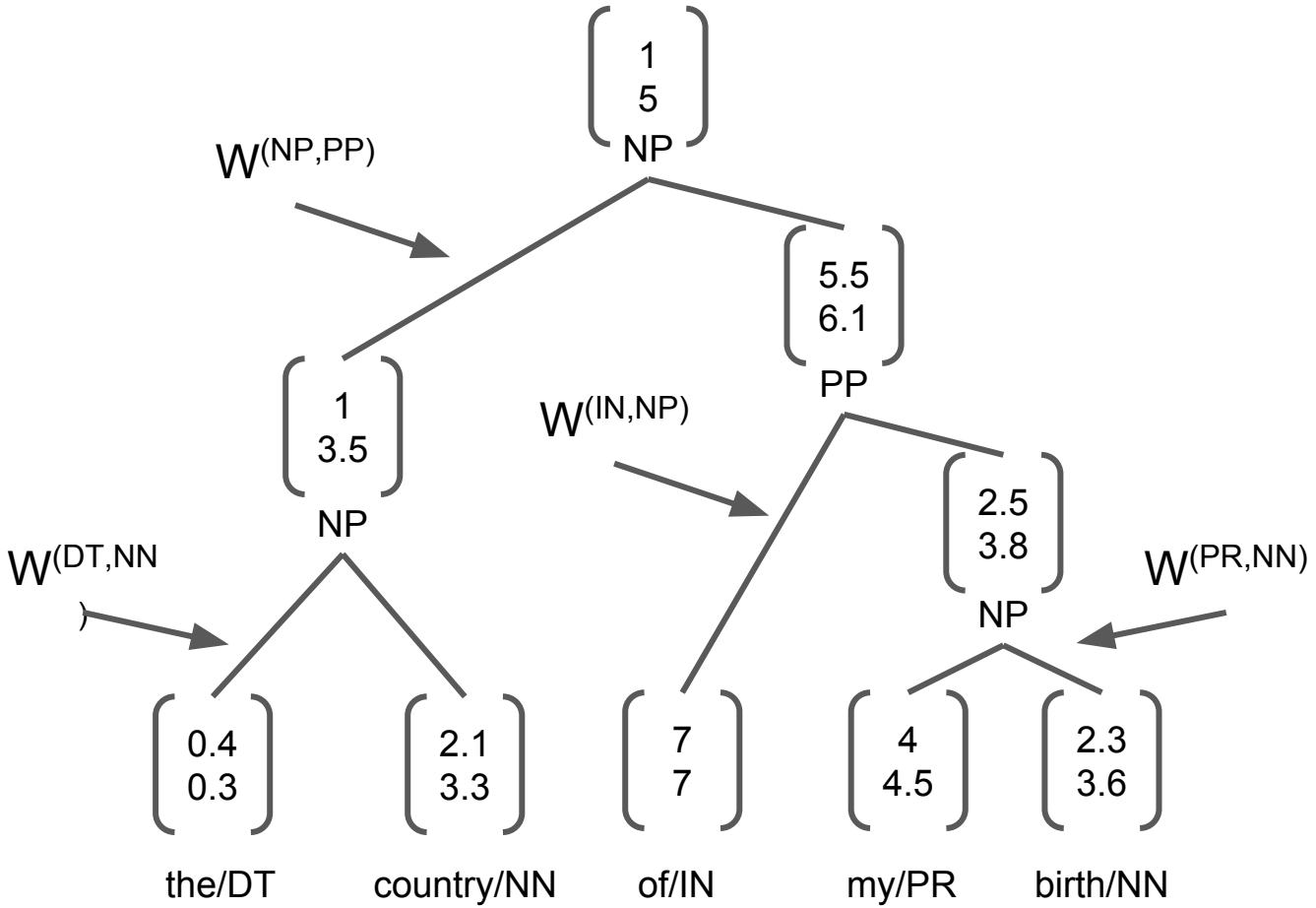
Los enfoques anteriores usan la misma matriz para todos los pares de nodos

- Misma función para todas las categorías sintácticas, puntuación, etc

Idea: Usar diferentes matrices W para dependiendo de las categorías de los hijos

- DT + NN usarán una matriz
- VP + NP usarán otra matriz
- …

A esto se le llama “desatar los pesos” (untie weights)


1
5
NP
W(NP,PP)
5.5
6.1
PP
1
3.5
NP
W(IN,NP)
2.5
3.8
NP
W(DT,NN
)
W(PR,NN)
0.4
0.3
2.1
3.3
7
7
4
4.5
2.3
3.6
the/DT
country/NN
of/IN
my/PR
birth/NN

Compositional Vector Grammar

Combinan RNN y PCFG

Hay una matriz de pesos distinta para combinar las categorías X e Y: $W^{(X,Y)}$

El vector de scoring también depende de las categorías: $v^{(X,Y)}$

Además, se usan las probabilidades de la PCFG

Compositional Vector Grammar

Sean el vector de la palabra a con categoría A y la palabra b con categoría B

El vector para la frase (a, b) se calcula como

$$p = f(W^{(B,C)}[b;c])$$

Y el score para esa frase es

$$s(p) = (v^{(B,C)})^T p + \log P(P_1 \rightarrow B\ C)$$

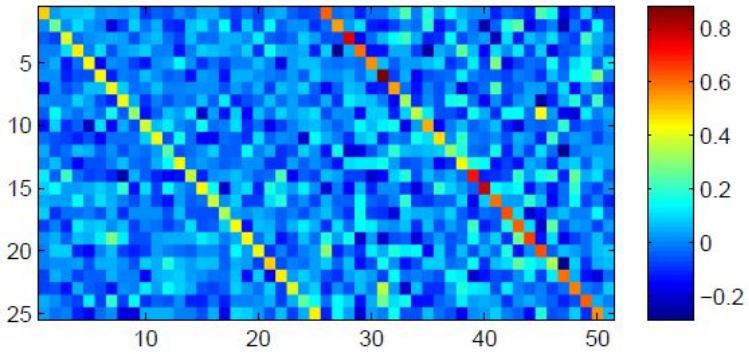
Donde P_1 es la categoría resultante combinar B y C

Compositional Vector Grammar

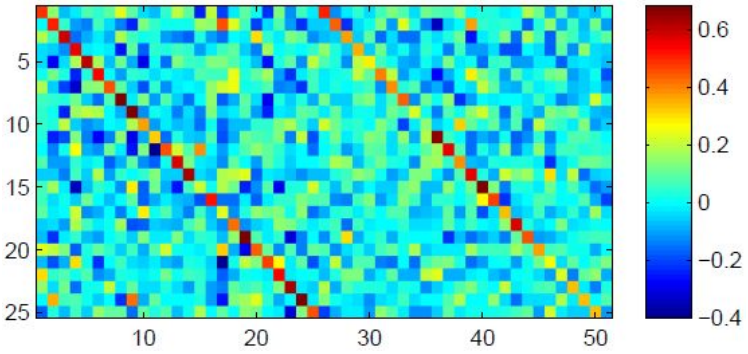
Resultados

Parser	dev (all)	test ≤ 40	test (all)
Stanford PCFG	85.8	86.2	85.5
Stanford Factored	87.4	87.2	86.6
Factored PCFGs	89.7	90.1	89.4
CVG (RNN)	85.7	85.1	85.0
CVG (SU-RNN)	91.2	91.1	90.4
Charniak-SelfTrain			91.0
Charniak-RS			**92.1**

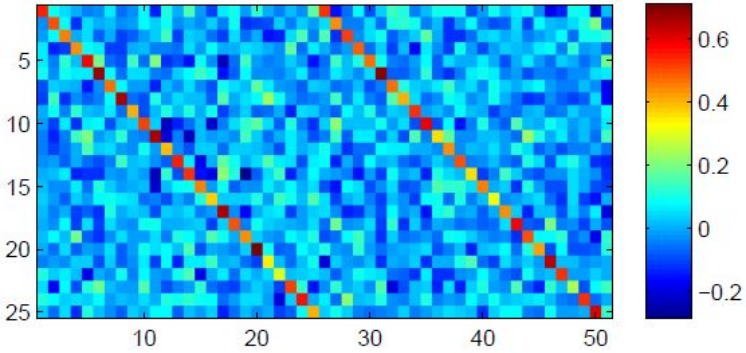
Compositional Vector Grammar



DT+NP



VP+NP



ADJP+NP

Compositional Vector Grammar

Reportan mejoras en PP attachment

Stanford factored parser:

- *[He [eats [spaghetti [with [a spoon]]]]]*
- *[He [eats [spaghetti [with meat]]]]*

Compositional Vector Grammar:

- *[He [eats spaghetti [with [a spoon]]]]*
- *[He [eats [spaghetti [with meat]]]]*

Análisis de sentimiento con RNN

Los phrase embeddings pueden usarse para análisis de sentimiento

- Calculamos el vector de una frase *p*
- Pasamos *p* por una capa softmax que elija la polaridad
- Entrenamos con un corpus apropiado

Esto significa que el phrase embedding debería capturar la polaridad de la frase

¿Es posible esto?

Análisis de sentimiento con RNN

Sea *x* el vector del adverbio “extremely”

Sea *n* el vector del adverbio “not”

¿Qué efectos deberían tener estos adverbios cuando se lo aplicamos a otra palabra *t*?

- Esperamos que *x* acentúe la polaridad de *t*
- Esperamos que *n* invierta la polaridad de *t* (o en algunos casos la atenúe)

Por ejemplo: f(*x*,*t*) = 2**w*, f(*n*,*t*) = -*t*

Análisis de sentimiento con RNN

Queremos: f(x,t) = 2*w, f(n,t) = -t

Pero calculamos los vectores resultantes usando una interpolación:

p_1 = tanh(W*[x;t])

p_2 = tanh(W*[n;t])

No importa cómo sea W, no hay forma de que esta interpolación lineal dé como resultado lo que queremos

Matrix-Vector Recursive Neural Network

Asocia a cada palabra no solo un vector, sino también una matriz

La palabra a se mapea en un vector *a* y una matriz *A*: $a \in R^n$, $A \in R^{n*n}$

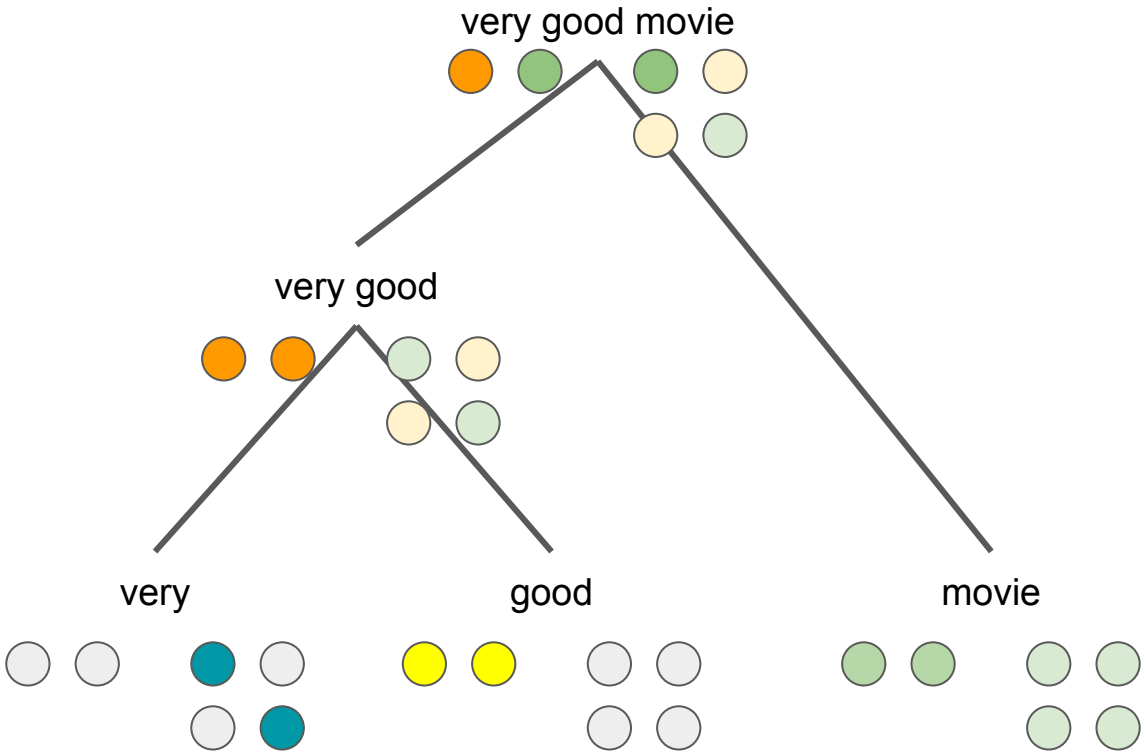
El vector de la frase de las palabras (*a*,*A*) y (*b*,*B*) usará producto de matrices:

$$p = \tanh(W[A*b;B*a])$$

Pero además cada frase tiene que tener también su matriz:

$$P = W_M(A,B)$$

Matrix-Vector Recursive Neural Network



Matrix-Vector Recursive Neural Network

Inicialmente los vectores son word embeddings entrenados de la manera habitual

Las matrices son la identidad + un poco de ruido aleatorio

Después del entrenamiento, se espera que aprenda cosas como:

- La matriz de “extremely” sería como la identidad multiplicada por un factor
- La matriz de “not” sería como la identidad negada
- Palabras que no modifican polaridad tendrían una matriz parecida a la identidad

Matrix-Vector Recursive Neural Network

Resultados

Método	Acc.
Tree-CRF (Nakagawa et al., 2010)	77.3
RAE (Socher et al., 2013)	77.7
Linear MVR	77.1
MV-RNN	**79.0**

Recurrent Neural Tensor Network

La red anterior mejora tiene problemas para resolver los siguientes casos

- Positivos negados: *“least X”* debería dar vuelta la polaridad de toda la frase
- Negativos negados: *“not bad”*, *“not dull”*, debería atenuar los negativos
- *“X but Y”*: debería ser positivo si el segundo es positivo

Se necesita un modelo más complejo para atacar estos casos

Recurrent Neural Tensor Network

$x \in R^{2d}$ es el vector unión de los vectores para las palabras a y b $\in R^{d}$

El vector de la frase [a b] es el vector h tal que:

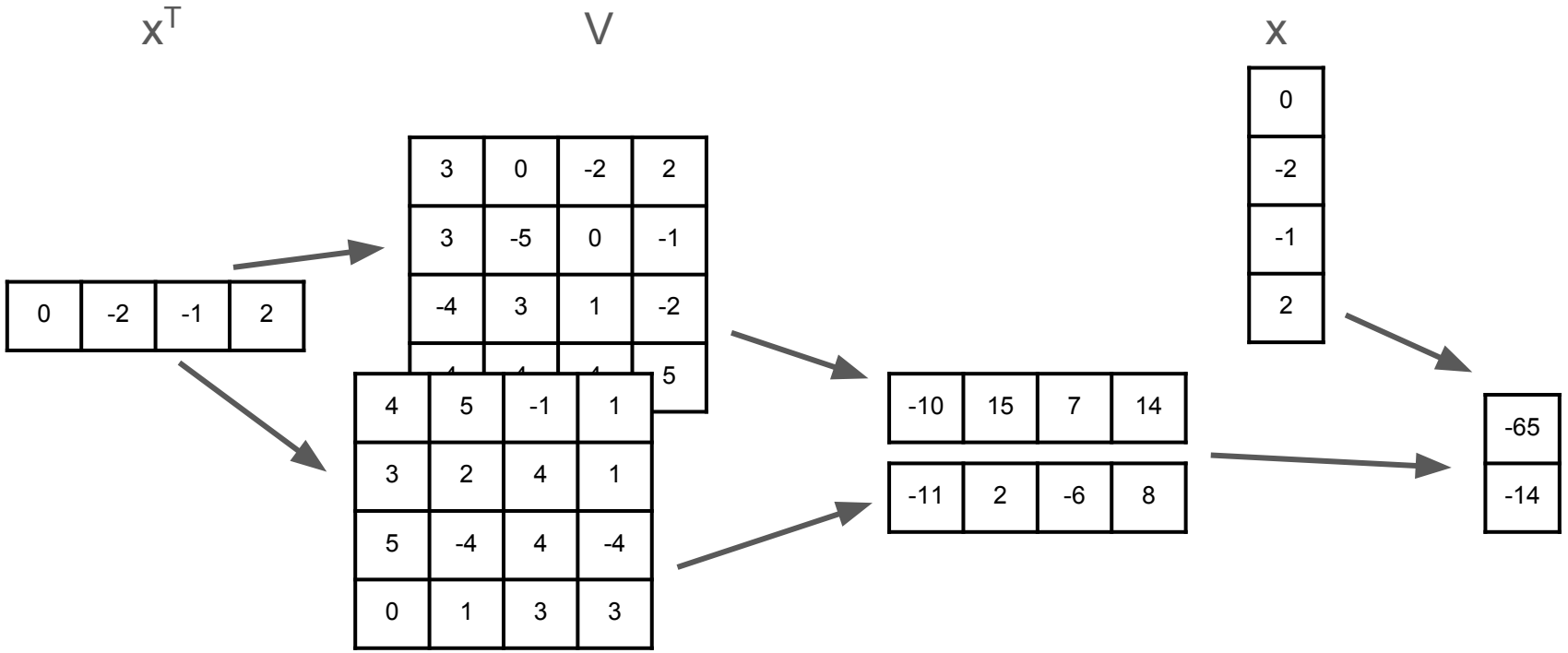
$$h = \tanh(x^{T}Vx + Wx)$$

$V \in R^{2d*2d*d}$

$W \in R^{2d*d}$

Se calcula $x^{T}V[i]x \ \forall i \in [1,2,...d]$ para formar un vector $\in R^{d}$

Recurrent Neural Tensor Network



Recurrent Neural Tensor Network

Hay que aprender los pesos de V y W

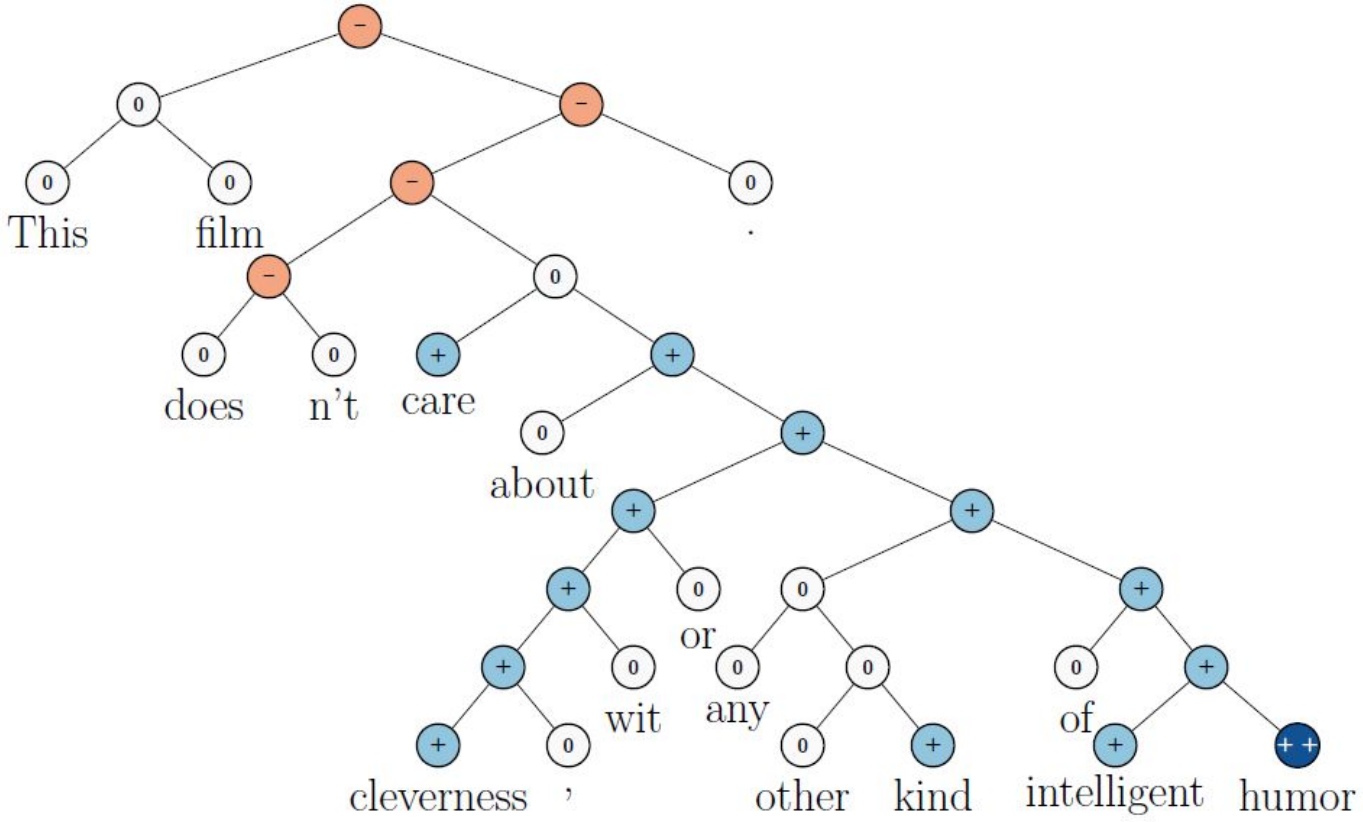
Usan un nuevo corpus anotado especialmente

11.855 oraciones

215.154 frases anotadas

Cada anotación indica un valor de polaridad de “very negative” a “very positive”, con varios grados

Recurrent Neural Tensor Network



Recurrent Neural Tensor Network

Entrenaron algunos de los modelos anteriores con este nuevo corpus y los resultados mejoran 2% o 3%

Pero con el modelo tensorial obtienen accuracy total de 85.4

Model	Negated Positive (Acc)	Negated Negative (Acc)
biNB	19.0	27.3
RNN	33.3	45.5
MV-RNN	52.4	54.6
RNTN	**71.4**	**81.8**

Referencias

Socher, R., Manning, C. D., & Ng, A. Y. (2010, December). Learning continuous phrase representations and syntactic parsing with recursive neural networks. In *Proceedings of the NIPS-2010 Deep Learning and Unsupervised Feature Learning Workshop* (pp. 1-9).

Socher, R., Bauer, J., Manning, C. D., & Ng, A. Y. (2013, August). Parsing with Compositional Vector Grammars. In *ACL (1)* (pp. 455-465).

Socher, R., Huval, B., Manning, C. D., & Ng, A. Y. (2012, July). Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning* (pp. 1201-1211). Association for Computational Linguistics.

Socher, R., Perelygin, A., Wu, J. Y., Chuang, J., Manning, C. D., Ng, A. Y., & Potts, C. (2013, October). Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)* (Vol. 1631, p. 1642).

Gracias!