Deep Learning for Answer Sentence Selection

Encontrar respuesta a las preguntas

Grupo L (Sebastián, Luciaon, Darío, Gabriel)

Agenda

1. Introducción
2. Trabajos relacionados
3. Modelo y experimentos
4. Resultados
5. Nuestras reflexiones

Agenda

1. Introducción
2. Trabajos relacionados
3. Modelos y experimentos
4. Resultados
5. Nuestras reflexiones

Introducción

- **¿Cuál es el desafío a resolver?**

Introducción

- ¿Cuál es el desafío a resolver?
- **¿Qué enfoques existen?**

Introducción

- ¿Cuál es el desafío a resolver?
- ¿Qué enfoques existen?
- **Propuesta del trabajo:** dominio abierto (TREC QA) y proyectar vectores

Agenda

1. Introducción
2. Trabajos relacionados
3. Modelo y experimentos
4. Resultados
5. Nuestras reflexiones

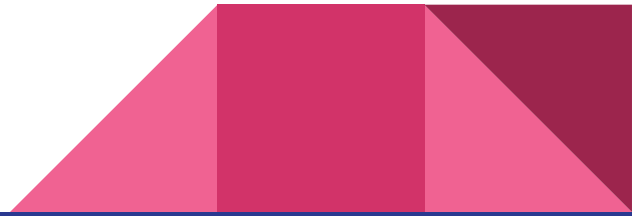
El modelo

Answer Sentence Selection como **problema de clasificación binaria**.

Dado **Q set de preguntas**, donde cada $q_i \in$ Q tiene una **lista de posibles oraciones de respuesta** asociada $\{a_{i1}, a_{i2}, \ldots, a_{im}\}$ junto a sus **juicios** $\{y_{i1}, y_{i2}, \ldots, y_{im}\}$ siendo y_{ij} = 1 si la respuesta es correcta y 0 en caso contrário.

Se tratan a estos datos como una **tripleta** (q_i, a_{ij}, y_{ij}) con el objetivo de aprender un clasificador sobre la misma y posteriormente predecir los juicios para nuevos valores QA.

Se asume que las respuestas a una pregunta tienen una alta similitud semántica respecto a sus preguntas.



¿En qué se diferencia de otros modelos similares?

Se descarta la medición de la similitud en base a información sintáctica y recursos semánticos artesanales.

En su lugar se opta por modelar preguntas y respuestas como **vectores** evaluando su similitud en un espacio vectorial.

Dada esta representación vectorial (R^d) de una pregunta **q** y una respuesta **a**, se tiene que:

$$p(y = 1|\mathbf{q}, \mathbf{a}) = \sigma(\mathbf{q}^T \mathbf{M} \mathbf{a} + b),$$

$$p(y = 1|\mathbf{q}, \mathbf{a}) = \sigma(\mathbf{q}^T \mathbf{M}\, \mathbf{a} + b),$$

M y **b** son parámetros del modelo.

Dada una candidata a respuesta, se genera una pregunta **q'** en base a la transformación **Ma**.

Luego se mide la similitud de la misma con la pregunta **q** mediante producto escalar.

Los valores se enmarcan luego en valores de probabilidad entre 0 y 1, lo que permite buscar el menor **cross entropy** posible sobre el los datos de entrenamiento ajustando parámetros y de dicha forma encontrar los valores ideales para la clasificación.

¿Cómo representar vectorialmente una sentencia en forma de vector?

Bag-of-words

- Remoción de Stop Words previamente.
- Dadas representaciones vectoriales de las palabras, se obtiene la representación vectorial de una sentencia mediante la suma de las representaciones de todas las palabras que la componen y la posterior normalización del largo de la sentencia.

$$\mathbf{s} = \frac{1}{|\mathbf{s}|} \sum_{i=1}^{|\mathbf{s}|} \mathbf{s}_i.$$

¿Cómo representar vectorialmente una sentencia en forma de vector?

Bag-of-words

Incapacidad de capturar elementos semánticos más complejos de una oración.

¿Cómo representar vectorialmente una sentencia en forma de vector?

Bigram-Model

Enfoque basado en Convolutional Neural Networks (CNN)

Se trata de un tipo de arquitectura de red neuronal caracterizado por el aprendizaje directo de los datos, sin requerir extracción manual de características.

Ha demostrado ser efectivo en tareas de predicción de sentimientos, semantic role labeling entre otros y es por ello que resulta prometedor su uso para este problema.

¿Cómo representar vectorialmente una sentencia en forma de vector?

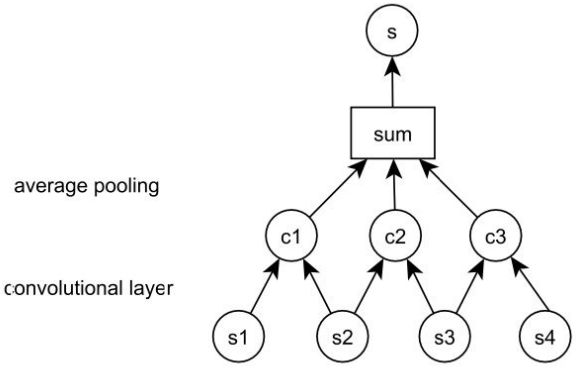
Bigram-Model

Convolutional Layer:

Identificación de características entre **bigramas** de una sentencia.

Average Pooling Layer:

Combina las características de todos los bigramas en un único vector.



¿Cómo representar vectorialmente una sentencia en forma de vector?

Bigram-Model

Convolutional Layer:

t es el vector convolucional compartido por todos los bigramas y se encarga de convertir cada bigrama en un valor ci.

$$\mathbf{c}_i = \tanh(\mathbf{t} \cdot \mathbf{s}_{i:i+1} + b).$$

¿Cómo representar vectorialmente una sentencia en forma de vector?

Bigram-Model

Finalmente obtenemos una representación vectorial de una sentencia de la misma dimensión que la representación vectorial de sus palabras (d-dimensional).

Dado que se trata de espacios vectoriales d-dimensionales, los bigramas así como t son matrices

La ecuación antes vista es calculada para cada fila de **t** y su correspondiente fila de **s**. Al igual que el proceso de average pooling es calculado para cada fila de la matriz de convolución.

¿Cómo representar vectorialmente una sentencia en forma de vector?

Bigram-Model

Aplicando las adaptaciones correspondientes obtenemos:

$$\mathbf{s} = \sum_{i=1}^{|\mathbf{s}|-1} \tanh(\mathbf{T}_L\, \mathbf{s}_i + \mathbf{T}_R\, \mathbf{s}_{i+1} + \mathbf{b}),$$

s_i es el vector de la i-esima palabra de s.

s el vector representación de la sentencia.

T_L y **T_R** parámetros del modelo en R^{dxd} y **b** el bias

Experimentos

Objetivo de los experimentos

1. **Comparar el modelo de los autores y el de trabajos previos**

2. **Comparar, en particular, la diferencia de rendimiento entre ambos modelos presentados**

Se utilizó el TREC Answer Selection Dataset

Descripción del Dataset

Lista de preguntas junto a una lista de respuestas asociadas a cada pregunta

¿Cómo se asociaron las respuestas a las preguntas?

Combinando la contabilización de palabras “overlapping” junto con la búsqueda de patrones

La correctitud de las respuestas fueron evaluadas manualmente

TREC Answer Selection

Data	# Questions	# QA Pairs	% Correct	Judgement
TRAIN-ALL	1,229	53,417	12.0	automatic
TRAIN	94	4,718	7.4	manual
DEV	82	1,148	19.3	manual
TEST	100	1,517	18.7	manual

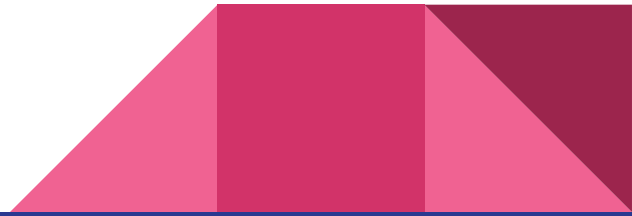
Mediciones

El objetivo es **clasificar las respuestas candidatas en base a su relación con la pregunta**

Se utilizó:

- Mean Average Precision (MAP)
- Mean Reciprocal Rank (MRR)

Dichos valores se calculan utilizando `trec_eval`



Configuración

- Se utilizaron word embeddings computados usando un neural language model ($d = 50$)
- Se inicializaron los pesos del modelo con valores randómicos utilizando una distribución de Gauss ($\mu = 0, \sigma = 0.01$)
- Los hiper parámetros fueron optimizados utilizando grid search sobre el puntaje MAP del dev set

Overlapping word count

Se agrega una segunda **feature** que cuenta la cantidad de palabras que, dado una tupla Q-A, están presentes tanto en la pregunta como en la respuesta

Tres finales features

1. Count
2. Count en base a IDF
3. Modelo distributivo

Agenda

1. Introducción
2. Trabajos relacionados
3. Modelo y experimentos
4. Resultados
5. Nuestras reflexiones

Resultados (bag-of-words vs. bigrama con CNN)

Model	MAP	MRR
TRAIN		
unigram	0.5387	0.6284
bigram	0.5476	0.6437
unigram + count	0.6889	0.7727
bigram + count	**0.7058**	**0.7800**
TRAIN-ALL		
unigram	0.5470	0.6329
bigram	0.5693	0.6613
unigram + count	0.6934	0.7677
bigram + count	**0.7113**	**0.7846**

Resultados (mejora con “+ count”)

1. **Q:** When did James Dean *die*?
   A1: In ⟨num⟩, actor James Dean was *killed* in a two-car collision near Cholame, Calif. (**correct**)
   A2: In ⟨num⟩, the studio asked him to become a technical adviser on Elia Kazan’s “East of Eden,” starring James Dean. (**incorrect**)
2. **Q:** How many members are there in the *singing group* the Wiggles?
   A1: The Wiggles are four effervescent *performers* from the Sydney area: Anthony Field, Murry Cook, Jeff Fatt and Greg Page. (**correct**)
   A2: Let’s now give a welcome to the Wiggles, a goofy new import from Australia. (**incorrect**)

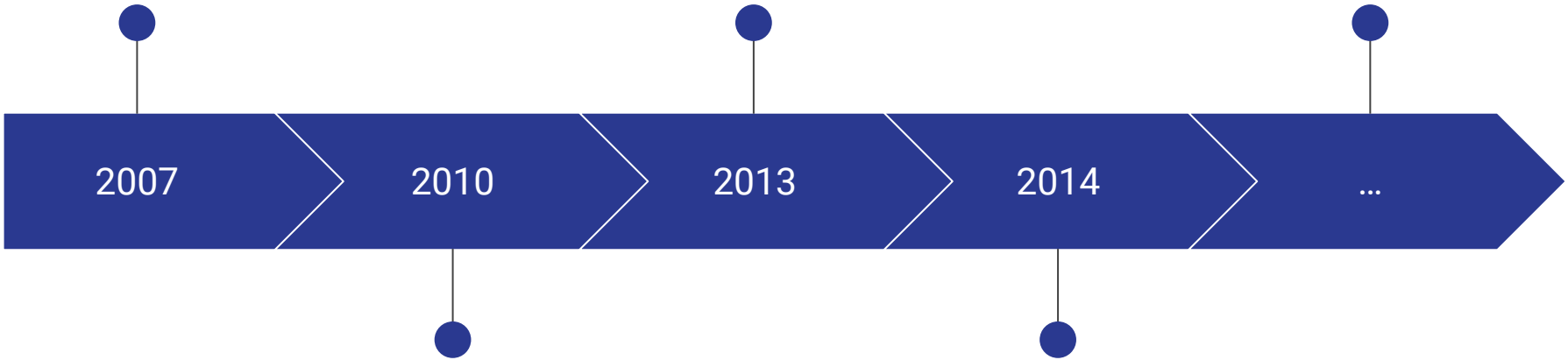
Resultados comparando estudios previos

System	MAP	MRR
Baselines		
Random	0.3965	0.4929
Word Count	0.5707	0.6266
Wgt Word Count	0.5961	0.6515
Published Models		
Wang et al. (2007)	0.6029	0.6852
Heilman and Smith (2010)	0.6091	0.6917
Wang and Manning (2010)	0.5951	0.6951
Yao et al. (2013)	0.6307	0.7477
Severyn and Moschitti (2013)	0.6781	0.7358
Yih et al. (2013) – LR	0.6818	0.7616
Yih et al. (2013) – BDT	0.6940	**0.7894**
Yih et al. (2013) – LCLR	**0.7092**	0.7700
Our Models		
TRAIN bigram + count	0.7058	0.7800
TRAIN-ALL bigram + count	**0.7113**	**0.7846**

MAP → 0.6029
MRR → 0.6852
(Wang et al.)

MAP → **0.7092**
MRR → **0.7894**
(Yih et al.)

¿?



MAP → 0.6091
MRR → 0.6917
(Heilman and Smith)

MAP → **0.7113**
MRR → **0.7846**
(Lei Yu - Karl Moritz Hermann - Phil Blunsom - Stephen Pulman)

Agenda

1. Introducción
2. Trabajos relacionados
3. Modelo y experimentos
4. Resultados
5. Nuestras reflexiones

Nuestras reflexiones

- Coincidimos en método más simple y flexible
- Tomar lo mejor de cada modelo puede traer mejoras
- Dependencia del word-embeddings

¡Muchas gracias!

¿Preguntas?

Grupo L (Sebastián, Luciaon, Darío, Gabriel)