DISEÑO E IMPLEMENTACIÓN DE UN ANALIZADOR DE DEPENDENCIAS PARA PROCESAMIENTO DE LENGUAJE NATURAL EN ESPAÑOL

Alejandro Alcalde [1]

[1] Grado Ingeniería Informática
Universidad de Granada

Índice

Soporte de Idiomas en Pipelines Actuales

ANNOTATOR	AR	ZH	EN	FR	DE	ES
Tokenize/Segment	✓	✓	✓	✓		✓
Sentence Split	✓	✓	✓	✓	✓	✓
Part of Speech	✓	✓	✓	✓	✓	✓
Lemma			✓			
Named Entities		✓	✓		✓	✓
Constituency Parsing	✓	✓	✓	✓	✓	✓
Dependency Parsing		✓	✓	✓	✓	
Sentiment Analysis			✓			
Mention Detection		✓	✓			
Coreference		✓	✓			
Open IE			✓			

Índice

Qué es el NLP

Definición

Ciencia que estudia la computación lingüística.

- Resúmenes.
- Traducción automática.
- Reconocimiento de voz.
- Sistemas de Diálogo Hablado.
- Clasificación de documentos.
- Análisis de sentimientos.

Objetivos

- Revisión bibliográfica.
- Elección de un parseador y diseño para Scala.
- Implementación y TDD.
- Evaluación y comparación de resultados.

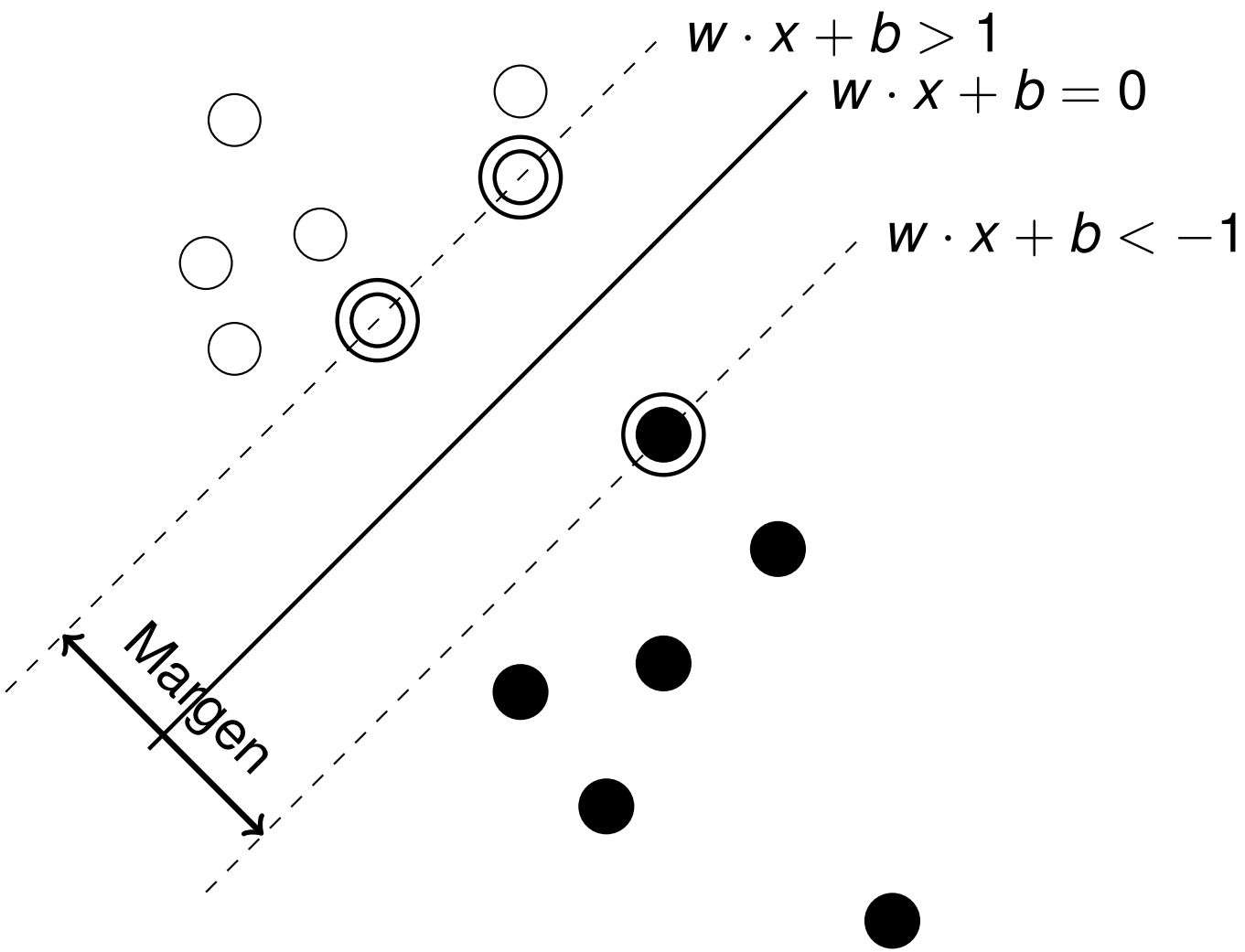
Índice

Algoritmo Seleccionado para Español

Statistical Dependency Analysis With SVMs

1: **Input Sentence:** $(w_1, p_1), (w_2, p_2), \cdots, (w_n, p_n)$
2: **Initialize:**
3: $i \leftarrow 1$
4: $\mathcal{T} \leftarrow \{(w_1, p_1), (w_2, p_2), \cdots, (w_n, p_n)\}$
5: no_construction $\leftarrow$ true
6: **while** $|\mathcal{T}| \geq 1$ **do**
7: **if** $i == |\mathcal{T}|$ **then**
8: **if** no_construction == true **then break**
9: **end if**
10: no_construction $\leftarrow$ true
11: $i \leftarrow 1$
12: **else**
13: **x** $\leftarrow$ getContextualFeatures($\mathcal{T}, i$)
14: $y \leftarrow$ estimateAction(model, **x**)
15: construction($\mathcal{T}, i, y$)
16: **if** $y ==$ Left or Right **then** no_construction $\leftarrow$ false
17: **end if**
18: **end if**
19: **end while**

SVMs
Qué es una SVM



SVMs
Ventajas de uso

- Gran poder de generalización.
- Con el *Kernel Trick* se combinan características.

Acción Desplazar

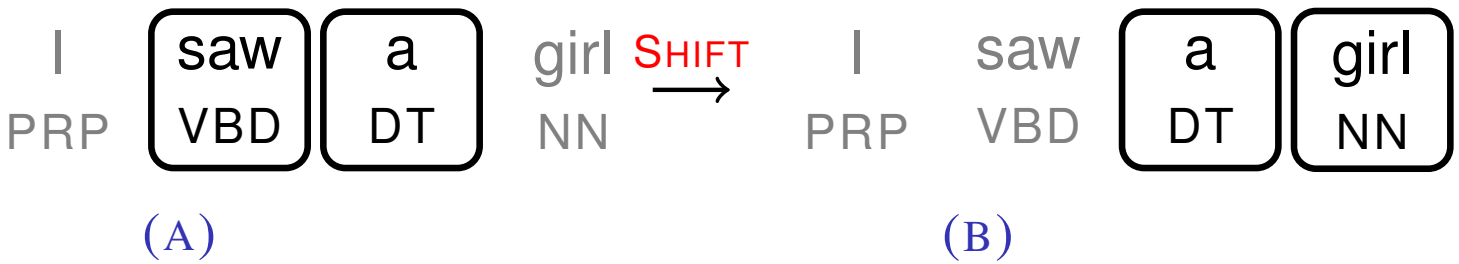


Figura 1: Desplazar. (a) Antes. (b) Después

Acción Derecha

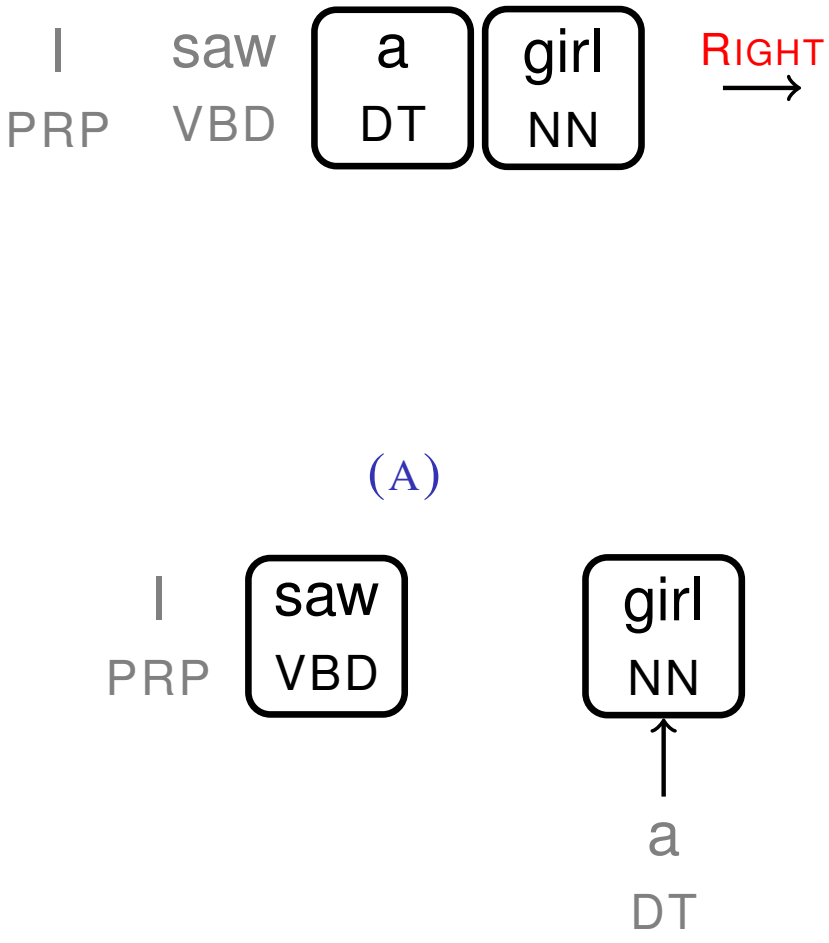


FIGURA 2: DERECHA. (a) Antes. (b) Después.

Acción Izquierda

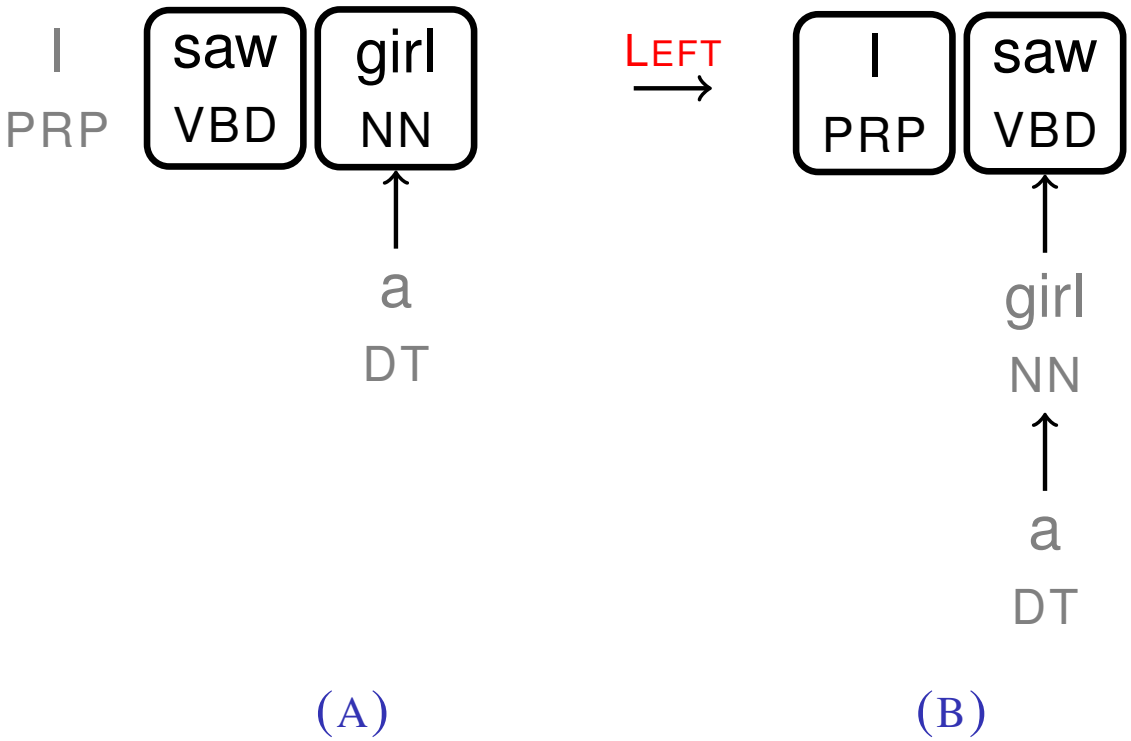
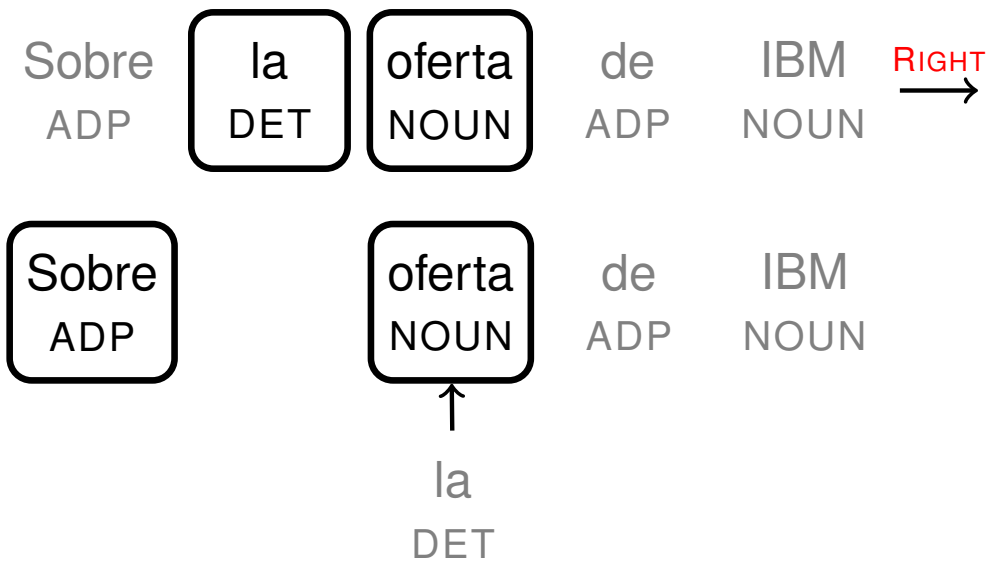


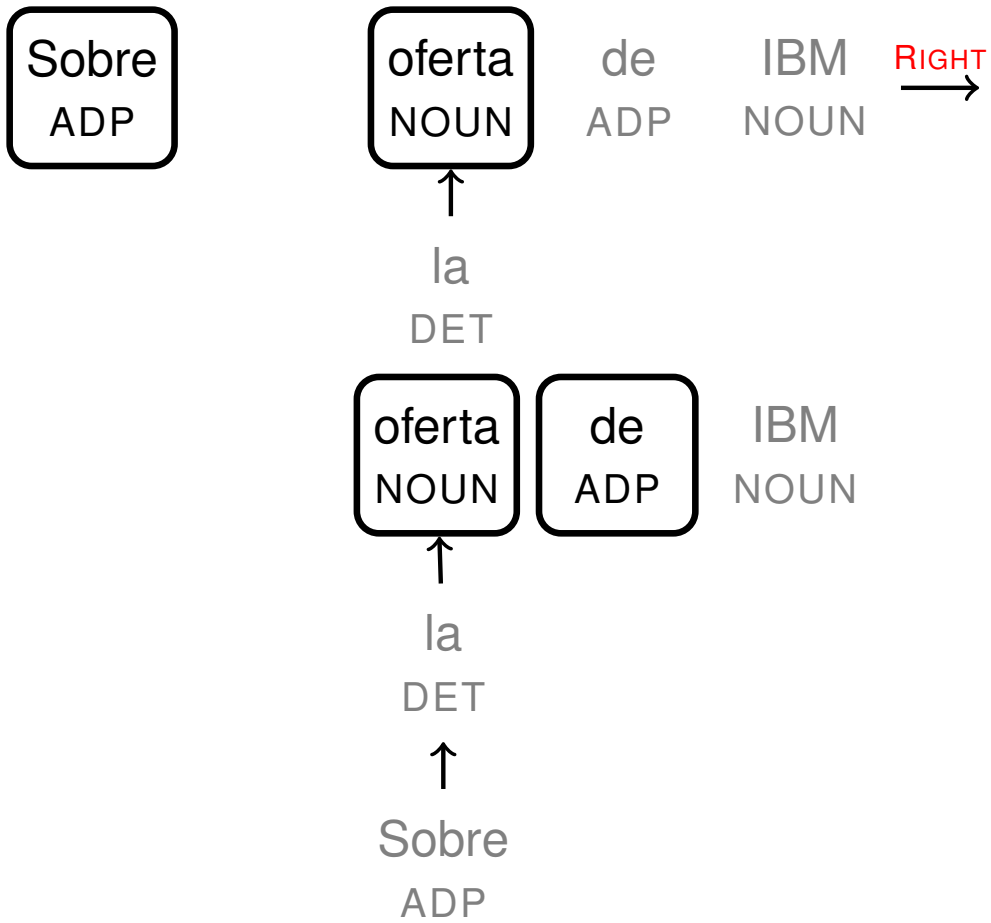
Figura 3: Izquierda. (a) Antes. (b) Después

EJEMPLO – "*Sobre la oferta de IBM*"

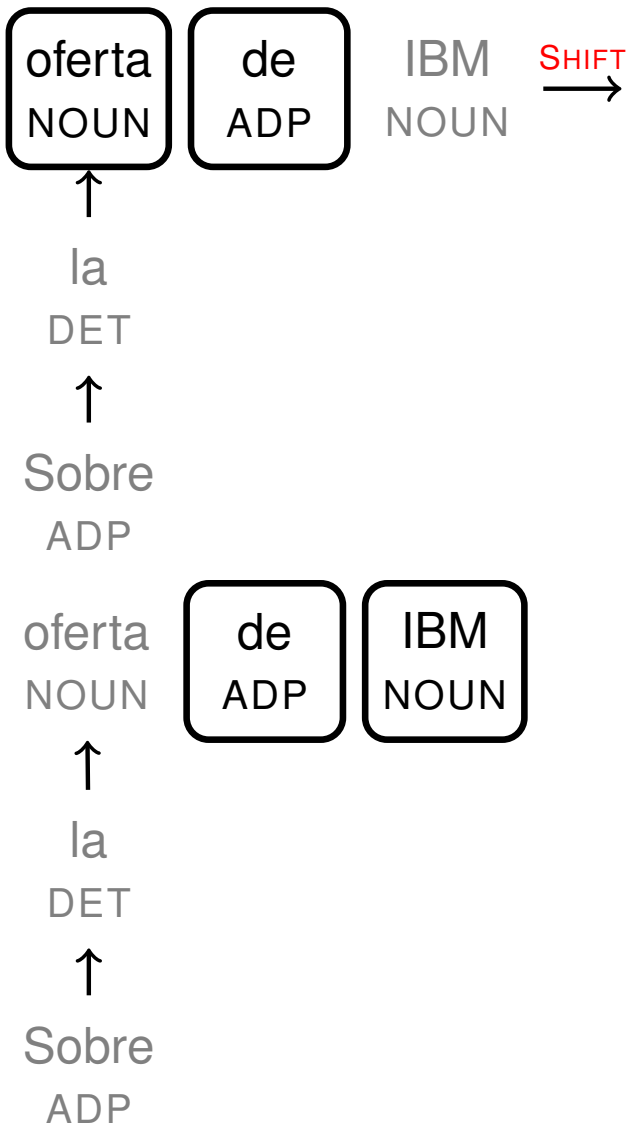
EJEMPLO – "*Sobre la oferta de IBM*"



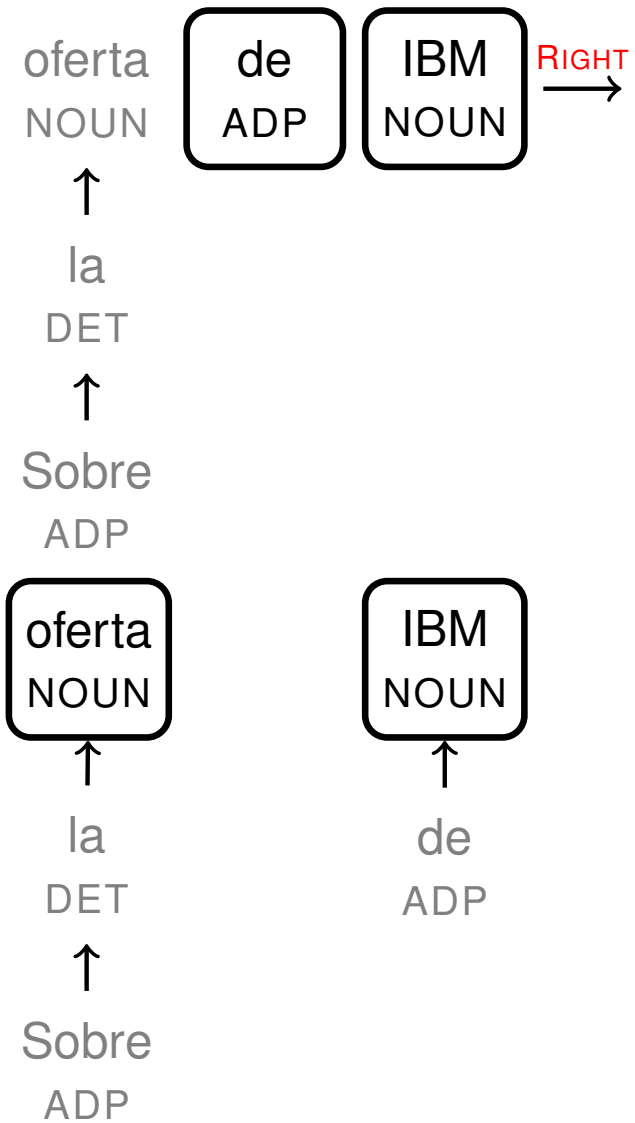
EJEMPLO – "*Sobre la oferta de IBM*"



EJEMPLO – "*Sobre la oferta de IBM*"



EJEMPLO – "*Sobre la oferta de IBM*"



Selección de Características

Definición

Una tripleta (p, k, v) donde:
p es la posición desde los nodos objetivo – offset –
k el tipo de caracteristica
v su valor.

Tipo	Valor
pos	POS *tag*
lex	La palabra
ch-L-pos	Nodo hijo modificando al padre por la izda.
ch-L-lex	Palabra del correspondiente ch-L-pos
ch-R-pos	Nodo hijo modificando al padre por la drcha
ch-R-lex	Palabra del correspondiente ch-R-pos

Índice

Resultados

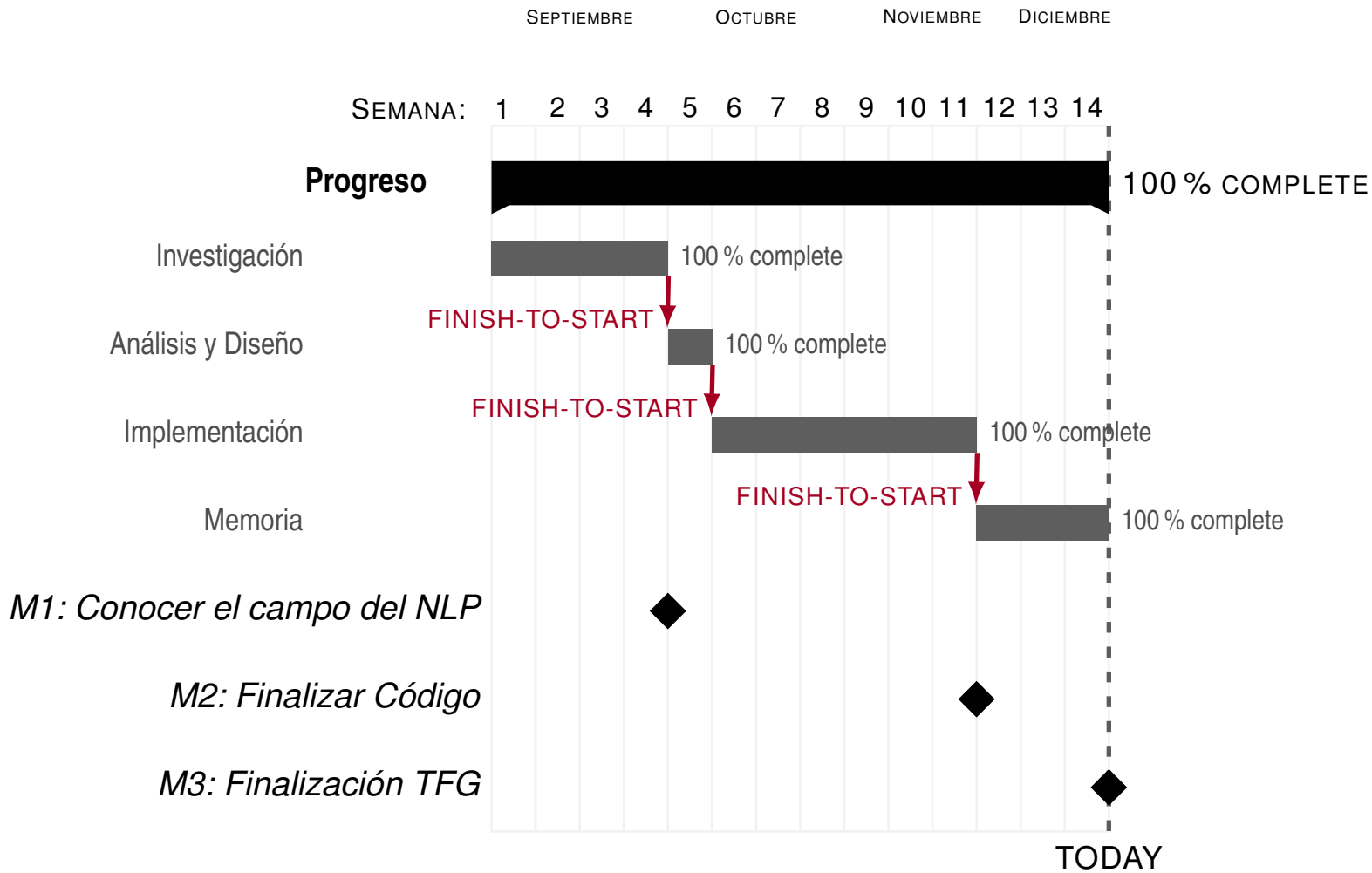
$$\textit{Dep. Acc} = \frac{\#\text{ correcto de padres}}{\#\text{ total de padres}}$$

$$\textit{Root Acc} = \frac{\#\text{ nodos raíz correctos}}{\#\text{ total de frases}}$$

$$\textit{Comp. Rate} = \frac{\#\text{ frases parseadas completamente}}{\#\text{ total de frases}}$$

Kernel: $(x' \cdot x'' + 1)^2$, Ctx: $(2, 4)$	TFG	ROHIT
Dep. Acc.	76 %	75 %
Root Acc.	67 %	70 %
Comp. Rate	15 %	11 %

Índice



Implementación
Por Qué en Scala

- Programación OO.
- Programación Funcional.
- Sintaxis breve.
- Escalable.
- Implementa algunos patrones.
- Traits.
- Amplio abanico reglas de visibilidad.

Implementación
Ventajas de Scala

Java

```
class WordCountMapper extends MapReduceBase
implements Mapper<IntWritable, Text, Text,
    ↪     IntWritable> {

  static final IntWritable one  = new
      ↪     IntWritable(1);
  // Value will be set in a non-thread-safe
      ↪     way!
  static final Text word = new Text;

  @Override
  public void map(IntWritable key, Text
      ↪     valueDocContents,
  OutputCollector<Text, IntWritable> output,
      ↪     Reporter reporter) {
    String[] tokens = valueDocContents
      .toString.split("\\s+");
    for (String wordString: tokens) {
      if (wordString.length > 0) {
        word.set(wordString.toLowerCase);
        output.collect(word, one);
      }
    }
  }
}

class WordCountReduce extends MapReduceBase
implements Reducer<Text, IntWritable, Text,
    ↪     IntWritable> {

  public void reduce(Text keyWord,
      ↪     java.util.Iterator<IntWritable>
      ↪     counts,
  OutputCollector<Text, IntWritable> output,
      ↪     Reporter reporter) {
    int totalCount = 0;
    while (counts.hasNext) {
      while (counts.hasNext) {
        totalCount += counts.next.get;
      }
      output.collect(keyWord, new
          ↪     IntWritable(totalCount));
    }
  }
```

Scala

```
class ScaldingWordCount(args : Args) extends
    ↪     Job(args) {
  TextLine(args("input"))
  .read
  .flatMap('line -> 'word) {
    line: String =>
      line.trim.toLowerCase.split("""\s+""")
  }
  .groupBy('word){ group => group.size('count)
      ↪     }
  .write(Tsv(args("output")))
}
```

Metodología: TDD y BDD

Test-Driven Development

- Rojo.
- Verde.
- Refactorizar.

Behavior-Driven Development

- *Given*.
- *When*.
- *Then*.

TDD para AA

Filosofía

Aplicar TDD a problemas de Aprendizaje

- R2 *Value*.
- ROC y AUC.
- Matriz confusión.
- Establecer un baseline.
- Intentar mejorarlo en cada iteración.

TDD para AA
Ejemplo


```
class DependencyParserCheckBaselineSpec extends Specification
  with GWT
  with StandardRegexStepParsers {def is = s2"""
  When training the model, set the following baselines ${featuresBaseline.start}
    Given Train data set: es_ancora-converted-train1
    Given Test data set: es_ancora-converted-test1
    When Genenaring Vocabulary
    Then Dep. Acc should be at least: 70%
    and Root Acc should be at least: 50%
    and Comp. Acc should be at least: 3%                ${featuresBaseline.end}
"""
// ...
```

Conclusiones y Vías Futuras

Conclusiones

- Implementado parseo para Castellano.
- Uso de SVMs.
- Scala.
- TDD para AA.

Vías futuras

- Más algoritmos.
- Código 100 % funcional.
- Más fases del Pipeline.
- Spark.

Gracias por su atención

algui91/NLP_Dependency_Parsing