Parsing de dependencias

Trabajo Práctico Final
Introducción al Procesamiento del Lenguaje Natural

Ramiro Camino, Guillermo Alejandro Gallardo Diez,
Martín Langberg, Fabrizio Borghini

Universidad de Buenos Aires

Segundo Cuatrimestre del 2014

Introducción

- Parsing de dependencias: cadena $\rightarrow$ árbol de dependencias.
- Vamos a estudiar los aportes de Joakim Nivre.

Definiciones: Grafo de dependencias (1)

- **string de entrada**: $w = w_1 \dots w_n$
- **intervalo**: $[i, j] = \{w_k : i \leq k \leq j\}$
- **grafo de dependencias**: $G = (V_w, E)$
  - $V_w = [1, n]$
  - $E \subseteq V_w \times V_w$
- **link de dependencia**: $(w_i, w_j) \in E$
  - w_i es el **padre** (o **cabeza**) de w_j
  - w_j es **hijo** de w_i

Definiciones: Grafo de dependencias (2)

- $(w_i, w_j) \in E$ si el link (w_i, w_j) existe
- $w_i \leftrightarrow w_j \in E$ si existe alguno de los links entre ambos tokens
- $w_i \rightarrow *w_j \in E$ si existe un camino dirigido de w_i a w_j
- $w_i \leftrightarrow *w_j \in E$ si existe alguno de los caminos dirigidos entre ambos tokens

Definiciones: Proyección de un nodo

- Conjunto de dependencias reflexivo transitivas.
- Conjunto de nodos a los que puede llegar.

$$\lfloor w_i \rfloor = \{w_j \in V : w_i \rightarrow *w_j\}$$

Definiciones: Bosque y Árbol de dependencias

- **Bosque de dependencias**: Grafo de dependencias que
  - es acíclico y
  - todo nodo posee un solo padre.
- **Árbol de dependencias**: Bosque con un solo nodo raíz.

Definiciones: Bosque proyectivo

- $\lfloor w_i \rfloor$ es un intervalo para cada palabra $w_i \in [1, n]$.
- $(w_i \leftrightarrow w_k \wedge w_i < w_j < w_k) \Rightarrow (w_i \rightarrow *w_j \vee w_k \rightarrow *w_j)$.

Definiciones: Grafo de dependencias bien formado

G **bien formado** $\Leftrightarrow$ G es un árbol de dependencias proyectivo.

Definiciones: Planaridad

- Sean (w_i, w_k), (w_j, w_l) links de dependencia:
  - sin pérdida de generalidad asumimos $min(i, k) \leq min(j, l)$,
  - son **links cruzados**
    $\Leftrightarrow min(i, k) \leq min(j, l) \leq max(i, k) \leq max(j, l)$.
- **Grafo de dependencias planar**: no posee links cruzados.

Definiciones: Multiplanaridad

Un grafo de dependencias $G = (V, E)$ es **m-planar** $\Leftrightarrow$ existen grafos de dependencias planares $G_1 = (V, E_1), \ldots, G_m = (V, E_m)$ tales que $E = \bigcup_{i=1}^{m} E_i$.

Definiciones Sistema de transiciones (1)

Representa un parser como un grafo donde:

- Los nodos son estados o configuraciones del parser.
- Las aristas son transiciones entre estados u operaciones elementales que cambian la configuración del parser.

Definiciones: Sistema de transiciones (2)

Formalmente es una tupla $S =< C, T, c_s, C_t >$ donde:

- C es un conjunto de posibles configuraciones.
- T es un conjunto de transiciones tales que $t \in T | t : C \rightarrow C$.
- c_s es una función que mapea cada posible entrada a una configuración. Es decir $c_s(w) \in C$.
- $C_t \subseteq C$ es un conjunto de configuraciones terminales.

Definiciones: Oráculo

Un **oráculo** es una función $o : C \to T$, que para cada configuración determina que transición realizar.

Inductive Dependency Parsing (Malt Parser)

- *Data-driven*
- Determinístico de orden lineal
- Se puede representar utilizando una tupla $< \Sigma_0, \Sigma_1, B, A >$ donde Σ_0 es la pila activa, Σ_1 la pila inactiva, B es el buffer de lectura y A el conjunto de dependencias que se derivan

Inductive Dependency Parsing (Malt Parser): Transiciones

- **Inicial**: $c_s(w_1 \dots w_n) =< [], [w_1 \dots w_n], \emptyset >$
- **Terminal**: $C_t = \{< \Sigma, [], A >\in C\}$
- **Transiciones**:
  - SHIFT: $< \Sigma, w_i|B, A > \rightarrow < \Sigma|w_i, B, A >$
  - REDUCE: $< \Sigma|w_i, B, A > \rightarrow < \Sigma, B, A >$
    solo si $\exists k|(w_k, w_j) \in A$
  - LEFT-ARC:
    $< \Sigma|w_i, w_j|B, A > \rightarrow < \Sigma, w_j|B, A \cup \{(w_j, w_i)\} >$
    solo si $\nexists k|(w_k, w_i) \in A$
  - RIGHT-ARC:
    $< \Sigma|w_i, w_j|B, A > \rightarrow < \Sigma|w_i|w_j, B, A \cup \{(w_i, w_j)\} >$
    solo si $\nexists k|(w_k, w_j) \in A$

Inductive Dependency Parsing (Malt Parser): Algoritmo de Covington

- Alternativa: Usar una lista en vez de una pila, es decir, poder enlazar cualquier token pasado con el siguiente token del input. Esto permite admitir dependencias no proyectivas, pero tiene un costo cuadrático en función del tamaño del input.

Inductive Dependency Parsing (Malt Parser): Dependencias pseudo-proyectivas

1. Cuando entrena, si encuentra un grafo no proyectivo, lo transforma moviendo hacia arriba los ejes cruzados hasta que no estén cruzados, y agrega anotaciones en los ejes para dejar constancia de esta transformación.
2. Cuando parsea trabaja como siempre.
3. Obtiene grafos de dependencia proyectivos pero con etiquetas enriquecidas.
4. Al final si el grafo tiene anotaciones trata de invertir la transformación para obtener un grafo no proyectivo.

Parser for 2-planar dependency structures

- Se pueden cubrir un 99 % de las frases en los treebanks más importantes utilizando grafos 2-planares al momento de generar las dependencias
- El parser se representa utilizando una tupla. $< \Sigma_0, \Sigma_1, B, A >$ donde Σ_0 es la pila activa, Σ_1 la pila inactiva, B es el buffer de lectura y A el conjunto de dependencias que se derivan.

Parser for 2-planar dependency structures: Transiciones

- **Inicial**: $c_s(w_1 \dots w_n) =< [], [], [w_1 \dots w_n], \emptyset >$
- **Terminal**: $C_t = \{< \Sigma_0, \Sigma_1, [], A > \in C\}$
- **Transiciones**:
  - SHIFT: $< \Sigma_0, \Sigma_1, w_i|B, A > \rightarrow < \Sigma_0|w_i, \Sigma_1|w_i, B, A >$
  - REDUCE: $< \Sigma_0|w_i, \Sigma_1, B, A > \rightarrow < \Sigma_0, \Sigma_1, B, A >$
  - LEFT-ARC: $< \Sigma_0|w_i, \Sigma_1, w_j|B, A > \rightarrow <$
    $\Sigma_0|w_i, \Sigma_1, w_j|B, A \cup \{(w_j, w_i)\} >$
    solo si $\nexists k|(w_k, w_i) \in A$ y no exista $w_i \leftrightarrow *w_j \in A$
  - RIGHT-ARC: $< \Sigma_0|w_i, \Sigma_1, w_j|B, A > \rightarrow <$
    $\Sigma_0|w_i, \Sigma_1, w_j|B, A \cup \{(w_i, w_j)\} >$
    solo si $\nexists k|(w_k, w_j) \in A$ y no exista $w_i \leftrightarrow *w_j \in A$
  - SWITCH: $< \Sigma_0, \Sigma_1, B, A > \rightarrow < \Sigma_1, \Sigma_0, B, A >$

Comparando Parsers: En teoría

- En el parser para estructuras de dependencias 2-planar las transiciones ARC y REDUCE se realizan solo en la pila activa.

Comparando Parsers: En teoría

- En el parser para estructuras de dependencias 2-planar las transiciones ARC y REDUCE se realizan solo en la pila activa.
- SHIFT saca el elemento del buffer al igual que antes pero ahora lo guarda en ambas pilas.

Comparando Parsers: En teoría

- En el parser para estructuras de dependencias 2-planar las transiciones ARC y REDUCE se realizan solo en la pila activa.
- SHIFT saca el elemento del buffer al igual que antes pero ahora lo guarda en ambas pilas.
- REDUCE ya no posee restricciones.

Comparando Parsers: En teoría

- En el parser para estructuras de dependencias 2-planar las transiciones ARC y REDUCE se realizan solo en la pila activa.
- SHIFT saca el elemento del buffer al igual que antes pero ahora lo guarda en ambas pilas.
- REDUCE ya no posee restricciones.
- ARC ya no saca la palabra de la pila al crear los arcos, y, a causa de los cambios descritos, se agrega a estas últimas transiciones una condición en la guarda que permite comprobar que no se produzcan ciclos antes de crear los arcos. Conserva la complejidad lineal.

Comparando Parsers: En teoría

- En el parser para estructuras de dependencias 2-planar las transiciones ARC y REDUCE se realizan solo en la pila activa.
- SHIFT saca el elemento del buffer al igual que antes pero ahora lo guarda en ambas pilas.
- REDUCE ya no posee restricciones.
- ARC ya no saca la palabra de la pila al crear los arcos, y, a causa de los cambios descritos, se agrega a estas últimas transiciones una condición en la guarda que permite comprobar que no se produzcan ciclos antes de crear los arcos. Conserva la complejidad lineal.
- El agregado de una nueva pila también hace que el 2-planar parser posea una nueva transición llamada SWITCH que se encarga de intercambiar las pilas, para indicar cuál es la activa.

Comparando Parsers: En teoría

- Malt parser va generando un grafo de dependencias basándose fuertemente en sus transiciones.

Comparando Parsers: En teoría

- Malt parser va generando un grafo de dependencias basándose fuertemente en sus transiciones.
- 2-planar parser va generando dos bosques de dependencias en cada pila evitando que existan pares de links cruzados dentro de las mismas, a la vez que permite que se crucen links pertenecientes a palabras en distintas pilas.

Comparando Parsers: Resultados empíricos

	Czech				Danish				German				Portuguese			
	LAS	UAS	NPP	NPR	LAS	UAS	NPP	NPR	LAS	UAS	NPP	NPR	LAS	UAS	NPP	NPR
2-P	79.2	85.3	68.9	60.7	83.8	88.5	66.7	20.0	86.5	88.8	57.1	45.8	87.0	90.8	82.8	33.8
M	79.8	85.7	76.7	56.1	83.6	88.5	41.7	25.0	85.7	88.6	58.1	40.7	87.0	90.6	83.3	46.2

Cuadro : Accuracy del parsing para el 2-planar parser (2P) comparado con Malt parser con transformaciones pseudo-proyectivas (M).
LAS = Labeled Attachment Score
UAS = Unlabeled Attachment Score
NPP = Precision on Non-Projective arcs
NPR = Recall on Non-Projective arcs.

Conclusiones

- El parser 2-planar es más simple que la implementación pseudo-proyectiva de Malt parser.

Conclusiones

- El parser 2-planar es más simple que la implementación pseudo-proyectiva de Malt parser.
- Define una clase concreta y bien conocida de estructuras parseables, mientras que no se sabe qué tipos de estructuras puede parsear el Malt parser.

Conclusiones

- El parser 2-planar es más simple que la implementación pseudo-proyectiva de Malt parser.
- Define una clase concreta y bien conocida de estructuras parseables, mientras que no se sabe qué tipos de estructuras puede parsear el Malt parser.
- Dado que las frases utilizadas en los corpus más importantes son 2-planares, podemos asegurar que el agregar una pila al parser genera otro parser más simple y robusto.

Conclusiones

- El parser 2-planar es más simple que la implementación pseudo-proyectiva de Malt parser.
- Define una clase concreta y bien conocida de estructuras parseables, mientras que no se sabe qué tipos de estructuras puede parsear el Malt parser.
- Dado que las frases utilizadas en los corpus más importantes son 2-planares, podemos asegurar que el agregar una pila al parser genera otro parser más simple y robusto.
- Agregar otra pila y hacer al parser 3-planar cubre el 100 % de las casos.

Preguntas?