Desambiguación de Palabras Polisémicas mediante Aprendizaje Semi-supervisado

Darío Garigliotti

Facultad de Matemática, Astronomía y Física
Universidad Nacional de Córdoba
dag0207@famaf.unc.edu.ar

Resumen Este trabajo es una exploración sistemática del impacto de diferentes aspectos de los algoritmos clásicos de desambiguación –no supervisada– de sentidos. Tras identificar los factores relevantes para su funcionamiento, muchos de los cuales estaban solamente implícitos en la descripción de estos algoritmos, implementamos una versión simplificada y levemente supervisada de un algoritmo clásico de reglas de decisión para desambiguación no supervisada. Evaluamos el impacto de cada uno de estos factores en el desempeño del mismo, entre ellos: el leve etiquetado inicial de ejemplos, la ecuación de confiabilidad de una regla y los criterios de aceptación de tales reglas de decisión. Los resultados obtenidos mediante una económica y poderosa evaluación con pseudo-palabras exhiben una performance aceptable en comparación con versiones muy optimizadas.

Keywords: Procesamiento de Lenguaje Natural, Desambiguación de sentidos, Aprendizaje automático semi-supervisado

1. Introducción y Motivación

La desambiguación de sentidos consiste en determinar, para una palabra polisémica –palabra que presenta el mismo significante o forma escrita y más de un significado–, qué sentido tiene la misma en una ocurrencia particular. La tarea de desambiguación de sentidos, más que una mejora, es una necesidad para que muchos procesos básicos de Procesamiento de Lenguaje Natural tengan buenos resultados, ya que implícitamente asumen que no hay ambigüedad. La estrategia dominante de estos algoritmos es la de emular los mecanismos humanos de desambiguación, esto es, obtener información del contexto de esa ocurrencia.

Este trabajo es una exploración sistemática del impacto de diferentes aspectos de los algoritmos clásicos de desambiguación –no supervisada– de sentidos. Hemos analizado estos algoritmos, identificando los factores relevantes para su funcionamiento –muchos de los cuales estaban solamente implícitos en la descripción de tales algoritmos– y valorando cuáles de ellos preservar y con qué criterios y parámetros particulares, para obtener así una implementación sencilla. La sección 3 presenta el desarrollo de un algoritmo levemente supervisado de reglas de decisión y la estrategia para evaluar el impacto de cada uno de estos factores en el desempeño del mismo. Los resultados obtenidos se detallan en la sección 4.

2. Trabajo Relevante

El problema de la desambiguación se ha abordado inicialmente con métodos de aprendizaje supervisado, necesitando para ello un gran *dataset* o conjunto de ejemplos –también llamados instancias o *contextos* donde ocurre la palabra polisémica– cada uno con una etiqueta de sentido desambiguado a mano.

Yarowsky, en [1], introduce un algoritmo de aprendizaje no supervisado que integra la estructura de listas de decisión con el etiquetado no supervisado de una buena cantidad inicial de ejemplos a través de determinar, para cada sentido de la palabra ambigua –palabra *target*–, una *colocación* o *evidencia semilla* que sea representativa de ese sentido. De esa manera, preserva la "no-supervisión" aunque la misma no es completa por requerir información de semilla, y es posible introducir cierto sesgo en la determinación de esas colocaciones iniciales. Además, usa dos heurísticas fundamentales: *un sentido por colocación* –las palabras que co-ocurren con la *target* dan fuertes indicios del sentido– y *un sentido por discurso* –puede asumirse que una palabra *target* tendrá el mismo sentido en múltiples ocurrencias dentro de un mismo discurso o artículo–. Con el trabajo de Abney ([2]) se inicia el análisis sistemático del algoritmo.

3. Métodos

3.1. Arquitectura del Sistema de Desambiguación de Sentidos

Dado un conjunto de instancias cada una con ocurrencia de una palabra *target* fija, el algoritmo de Yarowsky consiste en los siguientes pasos:

- Determinar colocaciones representativas y hacer el *training* inicial no supervisado. Particionar el conjunto de instancias en subconjuntos por etiquetas.
- Mientras el conjunto de instancias no-etiquetadas no converge:
  - Inspeccionando los ejemplos etiquetados, aprender reglas de decisión de la forma *colocación* $\Rightarrow$ *sentido*, aceptar las apropiadas y ordenar las aceptadas por criterio de log-verosimilitud –*confiabilidad* de la regla–.
  - Aplicar la lista de reglas a todo el conjunto, etiquetando un ejemplo con la primera regla de la lista que pueda aplicarse.
  - Aplicar el criterio de un sentido por discurso para filtrar ejemplos recientemente etiquetados.
- Tras converger a un conjunto residual estable de ejemplos no etiquetados, la lista final de reglas de decisión puede aplicarse sobre ejemplos aún no vistos.

El algoritmo presentado por Yarowsky en [1] abunda en criterios ligeramente justificados y/o aún entonces no parametrizados ni optimizados, y factores implícitos a determinar. Más precisamente, consideramos los siguientes aspectos y propusimos las respectivas decisiones de diseño para nuestra implementación:

- En nuestro trabajo, el *etiquetado inicial* –Factor 1– no se hace mediante tales colocaciones semilla sino desambiguando a mano 2 ejemplos por sentido por cada palabra *target*, convirtiéndolo así en un algoritmo semi-supervisado –esta decisión es clave, como se verá más adelante en los resultados–.

- ◇ Yarowsky considera algunos *tipos de colocaciones o evidencias* –Factor 2–, como las de co-ocurrencia adyacente al *target* de cualquier otra palabra, co-ocurrencia adyacente de palabras de contenido vs. palabras funcionales, y co-ocurrencia en una ventana de +/- n a m palabras; no puntualiza –ni valora en sus resultados– el impacto de cuáles usar –ni con qué valores concretos, por ejemplo en el caso de colocaciones de ventanas–. Usaremos, para obtener una versión sencilla, sólo colocaciones de co-ocurrencia simple, es decir, de la forma "tal palabra co-ocurre con la *target* en cualquier lugar del contexto".
- ◇ No implementamos la heurística de *un sentido por discurso* –Factor 3–.
- ◇ *La ecuación de confiabilidad o de probabilidad de una regla* –Factor 4– es distinta a la clásica de log-verosimilitud que propone Yarowsky en [1]. Dada $f(E_i)$ la cantidad de ejemplos en los que se presenta la evidencia o colocación E_i y $f(S_j, E_i)$ la cantidad de ejemplos etiquetados con el sentido S_j en los que se presenta E_i, la optimización que hacen Tsuruoka y Chikayama en [3] nos permite usar la siguiente ecuación de confiabilidad de que esa evidencia determine tal sentido:

$$(confiabilidad) = \frac{f(S_j, E_i)}{f(S_j, E_i) + f(\neg S_j, E_i)} = \frac{f(S_j, E_i)}{f|_{\{ejemplos\ etiquetados\}}(E_i)} \quad (1)$$

- ◇ El algoritmo presentado por Yarowsky en [1] no menciona explícitamente un *threshold* de confiabilidad –Factor 5– que deben superar las reglas para ser agregadas a la lista de decisión, mientras que en [2] Abney propone un *threshold* de $1/L$, con L = cantidad de sentidos a desambiguar. Estudios posteriores indican que se debería fijar uno mayor ya desde el caso $L = 2$. Fijamos inicialmente un *threshold* de 0.95.
- ◇ Factor 6: Mientras que en [1] se permite remover la etiqueta de un ejemplo al caer la regla directora debajo del *threshold*, [2] obliga a preservar siempre una etiqueta cuando se ha desambiguado por primera vez al ejemplo, aunque puede cambiar. En nuestro trabajo un ejemplo desambiguado permanece con la primera etiqueta: no se la puede cambiar ni volverlo a no-etiquetado.
- ◇ Cuando tienen la misma probabilidad, ordenamos las reglas a su vez por *cobertura* –Factor 7–, esto es la cantidad de ejemplos donde se presenta la colocación que dirige la regla –igual al denominador de (1)–. Este factor no es considerado por Yarowsky en [1] para el caso general, y sí lo hacen Tsuruoka y Chikayama en [3].

Se dispuso un plan de trabajo que consistía en la siguiente arquitectura:

1. Preprocesamiento del corpus y construcción del *dataset*;
2. Implementación del algoritmo de listas de decisión;
3. Evaluación *bananadoor*;
4. *Clustering* para contrastar performances.

Esta arquitectura de alto nivel y las subetapas realizadas, junto con las interfaces entre las mismas y las herramientas usadas en cada una se exponen en la Figura 1.

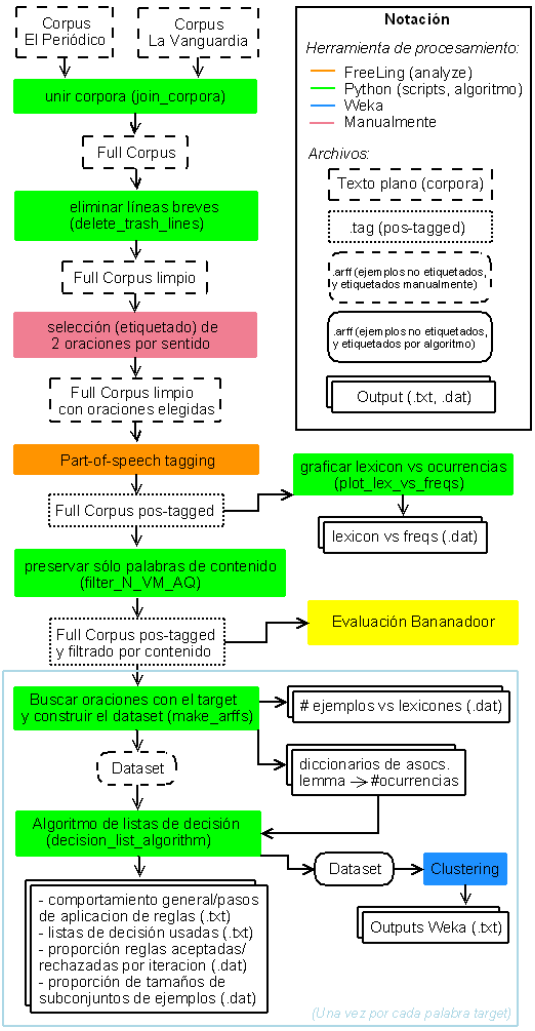


Figura 1. Arquitectura del sistema de desambiguación de sentidos semi-supervisado.

Tras determinar que cada ejemplo o unidad de contexto es una oración en la que ocurra cierta *target*, es importante destacar en esta arquitectura la tarea de lematización o normalización de palabras por su categoría gramatical, entendiendo que los accidentes morfológicos no alteran el sentido que sugieren.

Dispersión. También resulta fundamental el recorte de la enorme cantidad de palabras a un lexicón de lemas, dado en el lenguaje natural un conocido *fenómeno de dispersión*: el vocabulario del lenguaje es muy extenso y las palabras que ocurren en una oración son en general diferentes de las usadas en otra. En particular, tomando dos oraciones de un mismo discurso, una palabra en una de esas oraciones muy probablemente no ocurre en la otra, y más improbable –más disperso– aún cuantos más ejemplos se consideran.

Hacemos explícitos estos aspectos que claramente Yarowsky asume en [1] pero no ahonda, detallando los pasos que propusimos para el preprocesamiento en tal arquitectura:

1. Eliminación de toda línea de texto plano de menos de 10 palabras;
2. Etiquetado o desambiguación a mano de 2 oraciones por sentido por *target*;
3. Generación del archivo de lematización / Part of Speech - tagging –*pos-tagging*– mediante el uso de la herramienta FreeLing 2.2.2;
4. Filtrado del archivo de *pos-tagging* preservando sólo palabras de contenido, esto es, sustantivos –etiqueta morfológica de la forma N**** según el estándar parole / EAGLES–, verbos principales –VM****– y adjetivos calificativos –AQ****–;
5. *Splitting* de tal archivo de *pos-tagging* por oraciones –i.e. por contextos–;
6. Selección de aquellos contextos que contienen la palabra *target*;
7. Identificación de aquellos contextos desambiguados manualmente;
8. Definición del conjunto de todo lema distinto que aparece en esos contextos;
9. Construcción del lexicón: restricción del conjunto definido en punto 8 al conjunto de los lemas que aparecen en por lo menos 10 contextos;
10. Ordenamiento del lexicón por cantidad de contextos en los que ocurren;
11. Construcción del archivo de formato ARFF de los ejemplos de *training* para esa *target*, usando ese lexicón ordenado y los pocos etiquetados manuales identificados.

3.2. Arquitectura de Evaluación

La estrategia de evaluación *bananadoor* fue introducido por Schütze en [4] como un método sencillo y muy económico para la evaluación de algoritmos de desambiguación de sentidos. Consiste en elegir arbitrariamente dos palabras, por ejemplo "*banana*" y "*door*", y reemplazar en un corpus toda ocurrencia de cualquiera de las dos por la nueva pseudo-palabra *target* "*bananadoor*". El algoritmo de desambiguación se aplica sobre estas oraciones y se mide su performance: un ejemplo está bien desambiguado si el sentido que el algoritmo le da –sea "*banana*" o "*door*"– coincide con la palabra original que ocurría en esa oración y ha sido reemplazada por la pseudo-palabra. Aunque la ambigüedad de sentidos que se introduce mediante este método puede verse como artificiosa, el método tiene la ventaja de producir grandes cantidades de ejemplos para evaluaciones a gran escala, con un coste prácticamente nulo.

Este método de evaluación muestra también la independencia del algoritmo respecto al lenguaje natural sobre el que trabaja, en el sentido de asumir sólo el impacto de tales colocaciones y ninguna norma o convención del lenguaje.

Esta arquitectura de evaluación, diagramada en la Figura 2, guarda similitud con la arquitectura del sistema de la Figura 1 aunque incorpora las particularidades de los reemplazos antes explicados y parte de las decisiones de diseño del entorno experimental a ser usado, además de presentar dos algoritmos sencillos de desambiguación respecto a los cuales contrastar nuestra performance:

Algoritmo *Baseline*. Si la palabra que más se reemplazó por "*bananadoor*" fue, por ejemplo, "*door*" con un $k\,\%$ de los reemplazos, este algoritmo etiqueta

todos los ejemplos con el sentido "*door*" y obtiene $k\,\%$ de correctamente etiquetados.

Algoritmo *Random*. Con la información dada por el *baseline* –$k\,\%$ de los reemplazos es del sentido mayoritario–, etiquetar aleatoriamente dando tal sentido mayoritario un $k\,\%$ de las veces.

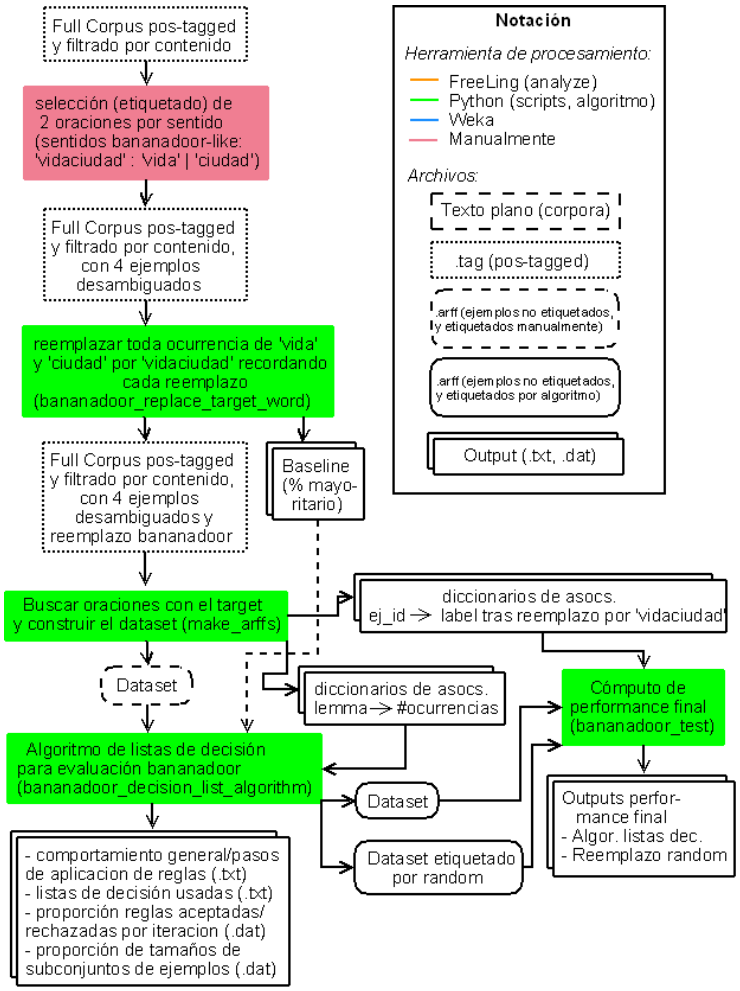


Figura 2. Arquitectura de la evaluación *bananadoor*.

4. Resultados

4.1. Entorno Experimental

Se trabajó con un corpus de 57 millones de palabras proveniente de las publicaciones digitales de los diarios españoles La Vanguardia y El Periódico de

Catalunya (resp. `http://www.lavanguardia.com/index.html` y `http://www.elperiodico.com/es/`). Por simplicidad, el problema se restringe a desambiguar palabras *target* de estas características:

- cada *target* tiene sólo 2 sentidos, en general muy distintos;
- todas son sustantivos, en particular, no se hace el caso donde adquiere diferentes sentidos según pueda pertenecer a diferentes categorías gramaticales;
- eventualmente, se incluye en polisemia al fenómeno similar de homonimia.

En una primera etapa de trabajo, se aplica la arquitectura de la Figura 1, obteniendo 5 *datasets*, uno por cada una de las palabras *target* elegidas, y se explora sobre ellos algunos aspectos conocidos de la lingüística (como la ley de Zipf) y sobre todo los particulares a investigar en nuestra implementación (impacto de los factores identificados en el desempeño, observando, por ejemplo, cantidad de iteraciones hasta converger, proporciones de subconjuntos del *dataset*, performances de *clustering*). El Cuadro 1 muestra estos *datasets*.

Cuadro 1. Las 5 *targets* binarias, y para cada una, las respectivas cantidades de ejemplos en su *dataset* y de lemas en su lexicón.

Target	Sentido A	Sentido B	Tamaño *Dataset*	Tamaño Lexicón
Manzana	Fruta	Superficie	712	144
Naturaleza	Índole	Entorno	2607	611
Movimiento	Cambio	Corriente	6509	1883
Tierra	Materia	Planeta	7874	2019
Interés	Finanzas	Curiosidad	14640	3104

Como segunda etapa, obtenemos un nuevo *dataset*, según la arquitectura de evaluación –reemplazo de toda "vida" o "ciudad" por "vidaciudad"–, de 62819 ejemplos y un lexicón de 3937 lemas que, a diferencia del recorte para los anteriores lexicones, acota razonablemente la cantidad de atributos del archivo ARFF tomando lemas que ocurren en por lo menos 30 oraciones. Esta reducción puede tener gran importancia si se somete al *dataset* a la técnica de *clustering* disponiendo de pocos recursos físicos de cómputo.

4.2. Análisis de Resultados

Aplicamos el algoritmo levemente-supervisado a cada *dataset*. Los resultados del Cuadro 2 muestran una rápida convergencia a un conjunto residual estable de ejemplos no-etiquetados. Claramente, es de impacto muy positivo cualquier factor que incremente la cantidad de reglas en la lista de decisión. Por ejemplo, hacer el etiquetado inicial –Factor 1– como en el algoritmo original nos da más reglas para la primera iteración, mientras que si se implementa el Factor 3 de un sentido por discurso cada iteración eventualmente entrega más ejemplos etiquetados a la siguiente; en ambos casos, para un *dataset* mucho mayor puede significar algunas iteraciones menos, aunque cada una requiere más tiempo para

inspeccionar justamente más reglas. En consecuencia, estos mismos factores, implementados según nuestras decisiones, tienen impacto negativo también en el *set* residual final. El Factor 2 es positivo para estos criterios de convergencia ya que más tipos de colocaciones capturan de manera más confiable algunas estructuras y convenciones que escapan a la co-ocurrencia simple.

Cuadro 2. Covergencia y proporciones de conjuntos residuales finales.

Targets / Cantidades	manzana	naturaleza	movimiento	tierra	interés	vidaciudad
N° de iteraciones necesarias para converger	5	6	7	8	7	6
% del *set* residual final respecto al *dataset*	6.46 %	33.10 %	42.83 %	18.97 %	33.25 %	77.62 %

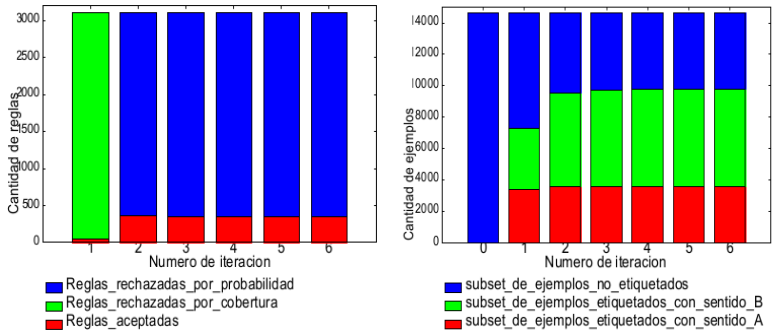
Nuestra decisión sobre el Factor 6 de no usar reetiquetado parece indicar un impacto positivo, sin embargo, sus variantes de uso son valoradas en los trabajos mencionados. Aparece así la cuestión de "velocidad versus precisión"; un factor como el de *threshold* –Factor 5– muy tolerante a reglas poco confiables puede impactar positivamente en la convergencia pero negativamente en cuán correcta es la desambiguación.

Como se ve en la última columna del Cuadro 2, un elemento ya explicado pero aún no identificado precisamente como factor es la *cota de lexicón*. Este factor resulta negativo en el desempeño para la etapa de evaluación desambiguando "vidaciudad": deja un gran *set* residual final pues exigimos que los lemas de su lexicón ocurran en por lo menos 30 contextos. Se mejora con la reducción de tal cota, pero hasta cierto valor ya que muy baja incorpora ruido por lemas infrecuentes. Aún no se observa impacto alguno del Factor 4 de confiabilidad, pero veamos los siguientes resultados.

La ecuación (1), según lo discutido por Tsuruoka y Chikayama en [3], puede producir estimaciones indeseadas, dada la escasa evidencia de la que se dispone al inicio. Como se observa en la Figura 3(a), la primera iteración del algoritmo acepta muy pocas reglas que entran ajustadamente en la cobertura exigida y rechaza la mayoría de ellas. La cobertura que exige nuestra versión es de al menos 1 evidencia: un criterio bastante razonable de aceptar reglas con al menos 2 evidencias nos resulta contraproducente pues con este criterio se observa que, debido al fenómeno de dispersión, el algoritmo no acepta ninguna regla en las primeras dos iteraciones y converge a un *dataset* sin nuevas etiquetas.

En resumen, el Factor 7 de cobertura resulta muy importante y sensible ya que una decisión apenas más estricta impide toda desambiguación. Tal factor está condicionado a nuestra forma mínimamente supervisada de hacer el primer *training*: ese Factor 1 tiene, por tanto, un impacto muy negativo en el desempeño. Como se ve en la Figura 3(b), esta lista inicial de reglas dispara un salto para luego refinarse con más cobertura y estabilizar las proporciones hasta converger.

El Factor 4 de ecuación impacta de manera particular, y en relación al factor de cobertura: usando una ecuación (5) –ver Apéndice B– de confiabilidad dada



(a) Proporción de reglas (b) Subconjuntos de ejemplos

Figura 3. Performance por iteración. *Target*: "interés".

por Tsuruoka y Chikayama en [3], que calcula el suavizado o *smoothing* óptimo para la confiabilidad dada en (1) –optimización que se hace para superar el problema de la poca cobertura inicial–, en la primera iteración del algoritmo las colocaciones de cobertura nula son la enorme mayoría, y adquieren una parte tal de la gran confiabilidad de los pocas colocaciones con alguna cobertura que todas las confiabilidades calculadas con (5) son menores a 0.001, es decir no superan nuestro *threshold* de aceptación ni ningún otro razonable.

En una segunda etapa, aplicamos la evaluación *bananadoor* y, como se observa en el Cuadro 3, la performance de 59.86 % de correctamente desambiguados supera sorprendentemente a los algoritmos de *Baseline* y de *Random*, y podemos decir que es un muy buen desempeño para nuestro algoritmo: la performance de una versión presentada en [3] que usa una ecuación de confiabilidad de log-verosimilitud y cobertura optimizada de al menos 3 evidencias es de un 69.4 %.

Cuadro 3. Performance de evaluación del algoritmo y performance de *clustering* (promedio de 5 *targets*).

	Evaluación *bananadoor*			*Clustering* (Promedio)	
Algoritmo	*Baseline*	*Random*	Listas de decisión	KMeans	EM (2 Clusters)
Performance	51.1 %	50.13 %	59.86 %	72.9534 %	72.49774 %

Por último, es superior la performance de correctamente "clustereados" –con la herramienta Weka 3.6.8– viendo el promedio de los resultados sobre cada uno de los 5 *datasets* de la primera etapa. Sin embargo, por limitaciones físicas, el *clustering* se ejecutó tomando sólo los 10 lemas más frecuentes de cada lexicón respectivo; de usarlo completo, especulamos que se introduce suficiente ruido para reducir su porcentaje.

5. Conclusiones y Propuestas de Trabajo Futuro

Hemos presentado una exploración sistemática de los elementos relevantes para el desempeño de un sistema de desambiguación de sentidos no supervisado.

Los resultados obtenidos indican que el *training* inicial es un factor decisivo para el desempeño del algoritmo, tanto en aspectos de convergencia como en la correctitud de la desambiguación. Tanto así que condiciona las decisiones a tomar respecto a otros factores de gran impacto como el *threshold* de confiabilidad y las restricciones de cobertura. También observamos desde los resultados que una optimización del *smoothing* sobre la ecuación de confiabilidad puede paradójicamente impactar muy negativamente en el desempeño, todo esto por las mismas consecuencias del etiquetado inicial.

Factores como reetiquetado, un sentido por discurso o más tipos de colocaciones pueden refinar la performance, a costo de una convergencia más tardía.

A partir de los resultados, algunas promisorias líneas de trabajo futuro son:

- Observar más precisamente el impacto de algunos de los factores recién mencionados. En particular, una versión que contemple también reglas por colocación o adyacencia de un lema de cierta categoría morfosintáctica, capturando situaciones habituales en algunos *targets* como por ejemplo la adyacencia a "naturaleza" de ciertos adjetivos: "*...propio de la naturaleza humana*".
- Introducir en la versión aquí presentada un criterio que exija una cobertura no fija sino cada vez más estricta a medida que progresan las iteraciones, para controlar así la población de reglas restringiéndolas a las más confiables.
- Observar los resultados con un *training* inicial que introduzca cierto sesgo contradictorio, esto es, contextos que tienen, cerca del *target*, colocaciones que deberían sugerir el otro sentido y no el indicado a mano, por ejemplo, en: "*...que la naturaleza sostiene lo* artificial...". Este sesgo, y en general cualquier otro –como elegir, al contrario, oraciones con colocaciones a priori muy representativas de cada sentido– en la etapa de desambiguación a mano es decisivo, aunque con nuestra estrategia de *training* inicial resulta un desafío no introducir al menos algo de sesgo.
- Realizar una etapa previa de inducción o descubrimiento de sentidos –por ejemplo, mediante *clustering* sobre el *dataset* original sin etiqueta alguna– y cotejar el desempeño desde un *training* inicial, como antes, a mano pero respecto a tal partición más "natural" de etiquetas. Esta propuesta es interesante porque aborda uno de los problemas clave para el éxito de la desambiguación como lo es la definición más apropiada del dominio de sentidos.

Referencias

1. Yarowsky, D.: Unsupervised word sense disambiguation rivaling supervised methods. Proc. of the 33rd Annual Meeting of the Association for Computational Linguistics, 189–196 (1995)
2. Abney, S.: Understanding the Yarowsky Algorithm. Computational Linguistics 30(3) (2004)
3. Tsuruoka, Y., Chikayama, T.: Estimating Reliability of Contextual Evidences in Decision-List Classifiers under Bayesian Learning. Proc. of the Sixth Natural Language Processing Pacific Rim Symposium, November 27-30, 2001 (2001)
4. Schütze, H.: Context space. AAAI Fall Symposium on Probabilistic Approaches to Natural Language, 113–120, Cambridge, MA (1992)

A. Pseudo-código del Algoritmo Original de Yarowsky

```
Dada una target w, con sus sentidos senses_set(w) = {sense_a, sense_b}
/* Parte 1 - Etiquetado inicial */
1-Identificar y etiquetar de forma no supervisada algunos ejemplos
iniciales, representativos de cada sentido de senses_set(w), para
partir el conjunto de ejemplos de training en 3:
  SET_A = etiquetados como A
  SET_B = etiquetados como B
  SET_? = no etiquetados
R = [] /* Lista de reglas de decisión */
converge = False /* Flag */
while( (no converge) Y (SET_? no vacio) ){ /* BEGIN WHILE */
  R_vieja = R
  /* Parte 2 - Construir lista de reglas, ordenadas según
  confiabilidad */
  R = []
  Para cada lema l_i en lista de lemas:
    2.1-Dado e_i = tipo de evidencia /* eg: l_i = 'mundo', luego
    e_i = "la palabra 'mundo' ocurre en algún lugar de la oración";
    extensible a reglas de adyacencia, de ventana de palabras, etc... */
    2.2-para cada sentido s en senses_set(w):
      Calcular c(s) = confiabilidad(s, e_i) /* confiabilidad de
      que esa evidencia e_i determine o implique el sentido s */
    2.3-calcular el valor s_i = sentido(e_i) = sentido s' que
    maximiza las confiabilidades {c(s')}
    2.4-crear regla r_i = (e_i -> s_i) y darle peso p_i = c(s_i)
    2.5-agregar (r_i, p_i) a la lista de reglas R
  Ordenar R descendiente segun pesos
  if R no es mejor que R_vieja:
    converge = True
  if no converge:
    /* Parte 3 - Aplicar la lista de decisión */
    Para cada oracion G en SET_?:
      3.1-etiquetar G según la regla de R más fuerte con la que
      matchee, si hay alguna
      3.2-mover G desde SET_? al set de etiquetados correspondiente
    /* Parte 4 - Un sentido por discurso */
    4-Opcionalmente, aplicar un sentido por discurso para ayudar
    a etiquetar, en particular, devolviendo a SET_? todas las
    instancias que presenten gran desacuerdo de sentido según
    compartan evidencia
} /* END WHILE */
/* Parte 4 - Un sentido por discurso */
4-Opcionalmente, aplicar un sentido por discurso para corregir
errores de labels
```

B. Optimización de Ecuación de Confiabilidad (Tsuruoka y Chikayama)

La ecuación con la que el algoritmo original de Yarowsky en [1] calcula la confiabilidad de una regla (Parte 2 del pseudo-código del Apéndice A) es:

$$(confiabilidad) = \log\left(\frac{P(S_j|E_i)}{P(\neg S_j|E_i)}\right) \tag{2}$$

donde E_i es la evidencia contextual o colocación y S_j es el sentido candidato para usar como etiqueta. En [3] se obtiene esta otra ecuación, que hace listas de decisión equivalentes a las de (2):

$$(confiabilidad) = P(S_j|E_i). \tag{3}$$

Desde allí puede verse que (3) puede estimarse –cuando hay un gran número de ejemplos con esa evidencia– por Máxima Verosimilitud con:

$$P(S_j|E_i) = \frac{f(S_j, E_i)}{f(E_i)} \tag{4}$$

con $f(E_i)$ la cantidad de ejemplos en los que se presenta la evidencia E_i y $f(S_j, E_i)$ la cantidad de ejemplos etiquetados con S_j en los que se presenta E_i.

La ecuación que usamos en este trabajo –y que sobreestima a (4) al restringir las cantidades del denominador al universo de sólo los etiquetados– es la (1):

$$(confiabilidad) = \frac{f(S_j, E_i)}{f(S_j, E_i) + f(\neg S_j, E_i)} = \frac{f(S_j, E_i)}{f|_{\{ejemplos\ etiquetados\}}(E_i)}$$

Se corrige la ecuación (1) en la implementación para que no haya divisor cero.

Usando aprendizaje Bayesiano, Tsuruoka y Chikayama obtienen en [3] una expresión de la distribución beta que involucra a $\Theta = P(S_j|E_i)$ y estiman así el mejor *smoothing*:

$$(confiabilidad) = E[\Theta] = \frac{f(S_j, E_i) + 1}{f(E_i) + 2} \tag{5}$$

***