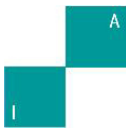
Inteligencia Artificial 47(2010), 38-53
doi: 10.4114/ia.v14i47.1570



INTELIGENCIA ARTIFICIAL

http://erevista.aepia.org/

Reagrupamiento en familias y lexematización automática independientes del idioma

Juan-Manuel Torres-Moreno
Laboratoire Informatique d'Avignon, UAPV
339 chemin des Meinajariès, BP 91228, 84911 Avignon Cedex 9, France
École Polytechnique de Montréal, DGI
CP 6079, succ. Centre-ville, Montréal (Québec) H3C 3A7, Canada
juan-manuel.torres@univ-avignon.fr

Abstract Este artículo presenta un sistema basado en métodos de regrupamiento no supervisado que detecta algorítmicamente las raíces o lexemas de familias morfológicas. La idea principal es la constitución de familias morfológicas a través de reagrupamientos iterativos. Los criterios de este reagrupamiento se basan en la similitud gráfica de las palabras, en su representación vectorial y en la correcta utilización de pares de sufijos (o firma de la familia) extraídos automáticamente. Las pruebas sobre corpora en francés, inglés y español muestran resultados muy interesantes en los tres idiomas, con una gran robustez e independencia del idioma.

Keywords: Statistical NLP (PLN estadístico), Statistical morphology (Morfología estadística), Grammatical Stemm (Raíz gramatical), Vector representation (Métodos vectoriales)

1. Introducción

Las palabras están compuestas por lexemas y morfemas. El lexema o raíz es la parte que no varía y que contiene su significado. El morfema es la parte variable, que se añade al lexema para completar su significado y formar nuevas palabras. De manera simplificada, una familia morfológica es un grupo de palabras relacionadas entre sí por un enlace morfológico de afijación. En la afijación se combinan una raíz y un afijo (prefijo o sufijo), ya sea para crear una nueva palabra (derivación) o bien para construir variantes de la misma (flexión). El análisis morfológico de las palabras es una fase muy importante en la construcción de sistemas de Procesamiento de Lenguaje Natural (PLN), porque tiene muchas aplicaciones en tareas como el resúmen automático de textos, la indexación de documentos, la clasificación textual y en sistemas de pregunta-respuesta a base de *queries*, entre otros [4]. Sin embargo, la realización de este análisis puede requerir el uso de recursos externos (como diccionarios, analizadores, reglas, etc.) que pueden ser caros, difíciles de construir y demasiado dependientes de un idioma o de un dominio específico [34]. Un ejemplo de este análisis es la lematización de palabras, que permite reducir la dimensión del espacio vectorial de representación (es decir, el léxico) en los sistemas de búsqueda y extracción de información [4, 34].

Este artículo ofrece un nuevo algoritmo de adquisición estadística de familias morfológicas, capaz de obtener su lexema, evitando el uso de recursos externos o el conocimiento *a priori* de una lengua. Para ello se formula la adquisición de familias morfológicas como un problema de clasificación no supervisada. Es decir, el objetivo es organizar un conjunto de datos en grupos homogéneos y contrastados: en nuestro caso, los grupos son familias de palabras morfológicamente relacionadas. El método propuesto tiene

ISSN: 1988-3064(on-line)

como entrada únicamente un conjunto de palabras que serán reagrupadas en familias y calcula el lexema asociado. A diferencia de los sistemas clásicos de aprendizaje supervisado, este algoritmo no necesita ajustar parámetros, por lo que no requiere un conjunto de entrenamiento. Otra de las características importantes de nuestro algoritmo es que no está ligado a un idioma o a un tema particular. De esta manera, el contexto de cada palabra no será utilizado en el regrupamiento.

Este artículo está organizado como sigue: en la sección 2 se ofrece una revisión bibliográfica de los métodos utilizados y de sus aplicaciones potenciales. En la sección 3 se presenta un algoritmo *naïf* de cálculo probabilístico de lexemas, que será mejorado en la sección 4, con un sistema basado en una representación vectorial combinada con la noción de firma (pares de sufijos característicos de una familia) de las familias morfológicas. El algoritmo es evaluado sobre corpora en tres idiomas (inglés, español y francés) en la sección 5. Finalmente, en la sección 6 se presenta una discusión, las conclusiones y algunas perspectivas de nuestro trabajo.

2. Estado del arte

Una gran cantidad de problemas interesantes de búsqueda de información como son el Resumen Automático de texto (RA), la Indexación de Documentos (ID), la Clasificación Textual (CT) y los sistemas de Preguntas-Respuestas (QA), entre otros pueden ser resueltos con algoritmos de PLN. Parece ser evidente que el avance futuro en estas áreas se dará a través de una mejor comprensión automática de la lengua. Sin embargo, el estado actual de las investigaciones está lejos de lograr esta comprensión y presenta numerosas dificultades en todos los niveles del análisis del documento escrito. Los problemas son diversos y pueden darse a nivel morfológico, sintáctico, semántico o pragmático [34]. Los tres últimos están fuera de alcance de este artículo y no serán estudiados. Nos interesaremos únicamente en el aspecto morfológico y haremos enfásis en un procesamiento que utiliza exclusivamente técnicas estadísticas.

La morfología estudia la estructura interna de las palabras para delimitar, definir y clasificar sus unidades, las clases de palabras a las que da lugar (morfología flexiva) y la formación de nuevas palabras (morfología léxica). Es decir, estudia la forma de las palabras a partir de su flexión (indicaciones de los casos, el género, el número, el modo, el tiempo, etc.), la derivación (prefijos, sufijos, infijos) y la composición (palabras compuestas). Por su parte, la morfosintaxis estudia las reglas de combinación de los morfemas (unidades mínimas con significado), en función de la configuración sintáctica de la oración. En la práctica, dentro del PLN, el análisis morfológico consiste en segmentar un texto en sus unidades elementales[1] y en identificar las diferentes características de estas unidades. La morfosintaxis se refiere al conjunto de elementos y reglas que permiten construir oraciones con sentido y carentes de ambigüedad mediante el marcaje de relaciones gramaticales, concordancias, indexaciones y estructura jerárquica de los constituyentes sintácticos.

La lematización [34] es un proceso que actúa en el nivel morfológico del PLN. Hay dos tipos de lematización: la lematización sin contexto, cuando se tiene sólo la forma de una palabra sin su contexto (como en un diccionario), y la lematización generada en el contexto [40]. En este último caso se puede obtener mayor información morfosintáctica, por ejemplo, la categoría gramatical, y eliminar así la ambigüedad en los homógrafos. En este artículo nos referiremos únicamente a la lematización sin contexto. La lematización intenta normalizar automáticamente los términos pertenecientes a una misma familia, próximos en significado, reduciéndolos a una forma canónica o lema. De este modo los sustantivos son transformados al masculino singular y los verbos al infinitivo. El objetivo principal es obtener, con un número mínimo de caracteres, el máximo de información del término en cuestión. Hay que considerar que, si la palabra no puede ser lematizada; como es el caso típico de los nombres propios, las palabras en idioma extranjero o las palabras desconocidas (*Out Of Vocabulary, OOV*), los lematizadores no pueden asociarle ninguna información. La lematización supone entonces que un análisis morfosintáctico ha sido realizado previamente. Las flexiones son las modificaciones realizadas en el lema para distinguir las formas de conjugación (persona, tiempo, modo, voz, flexión verbal) o de género, número y caso (flexión nominal). En inglés, las flexiones no son muy numerosas y son relativamente fáciles de encontrar a partir del lema y de la categoría gramatical. Por el contrario, en las lenguas latinas como el italiano, el francés, el español, el catalán o el portugués, la tarea resulta ser mucho más difícil [13]. El proceso de *stemming*

[1] En inglés *tokenisation*.

[28, 33, 36, 41] consiste en truncar las palabras en sus radicales después de la supresión de los afijos. La lematización y la extracción de raíces son dos técnicas de normalización comúnmente utilizadas en PLN y permiten reducir la cantidad de términos a procesar.

2.1. Algoritmos de *stemming* y de lematización

Los métodos para el análisis morfológico pueden ser muy variados. En [21] y [22] el lector puede hallar un estado del arte muy completo al respecto. Ejemplos de este tipo de análisis son la comparación de grafos [20], la utilización de n-gramas [16, 34], la búsqueda de analogías [32], los modelos superficiales a base de reglas [38, 31], los modelos probabilísticos [12], la segmentación por optimización [11, 19], el aprendizaje no supervisado de las familias morfológicas por clasificación jerárquica ascendente [7], la lematización usando distancias de Levenshtein [14] o la identificación de sufijos por medio de la entropía [42]. Estos métodos se distinguen por el tipo de resultados obtenidos, ya sea la identificación de lemas, *stems* o sufijos. Por ejemplo, el sistema FLEMM[2] [15] es un analizador flexional para la lengua francesa que necesita como entrada un texto previamente etiquetado por alguno de los dos sistemas, WINBRILL[3] o TREETAGGER[4]. FLEMM produce, entre otros resultados, el lema de cada palabra del texto de entrada. El sistema está compuesto por un módulo central que realiza la lematización de cada palabra a través de los siguientes pasos: *i*) Descomposición de la palabra de acuerdo con la terminación más probable; *ii*) Acoplamiento de la base obtenida mediante reglas de reajuste para reducir la alomorfia; *iii*) Si es necesario, el cálculo de la terminación en infinitivo. En cada paso se recupera la información flexional calculada sin ambigüedad. La activación de ciertas reglas está sujeta a la consulta previa de la lista de excepciones apropiada. TREETAGGER [25] es otra herramienta muy popular que permite anotar textos con informaciones de las *Parts-Of-Speech* (POS) (los tipos de palabras son, por ejemplo, sustantivos, verbos, infinitivos y partículas) y con información de la lematización. Desarrollado en el *Institute for Computational Linguistics* de la *University of Stuttgart*, TREETAGGER permite el etiquetado del alemán, inglés, francés, italiano, holandés, español, búlgaro ó ruso, entre otros idiomas. TREETAGGER hace uso de aprendizaje automático supervisado (en particular de árboles de decisión [9, 39] y de métodos probabilísticos). Puede ser adaptable a otros idiomas siempre y cuando los recursos léxicos y los corpus etiquetados manualmente se encuentren disponibles.

Los métodos de *stemming* y de lematización pueden ser aplicados cuando los términos son morfológicamente similares. Pero cuando la similitud es semántica, deben usarse métodos de búsqueda léxica. Para reducir la variación semántica, muchos sistemas emplean diccionarios o tesauros para relacionar dos palabras con formas completamente diferentes [37]. Los dos procedimientos son complementarios ya que el *stemming* verifica las similitudes a nivel de la grafía para inferir proximidad lexical, mientras que los algoritmos léxicos usan datos terminográficos con enlaces a sinónimos [29]. Para el ruso y el español, [18] presentan un sistema basado en la transformación estática de los *stem* alomorfos y un "análisis a través de la generación". Esto permite a los autores el uso de los modelos morfológicos orientados a la generación, en lugar de desarrollar modelos de análisis especiales. En [17] se presenta un algoritmo no supervisado para el *stemming* de lenguas flexionales. Según los autores, el algoritmo podría aplicarse a lenguajes aglutinantes, con las modificaciones adecuadas. Se usan algoritmos genéticos para aproximar las soluciones.

Trabajos como el de [46] proponen la aplicación de familias morfológicas fusionadas en un solo término para reducir la variedad lingüística de documentos indexados escritos en español. Presentan un sistema de generación automática de familias morfológicas, usando morfología derivativa productiva. Este sistema utiliza un mínimo de recursos lingüísticos e implica un bajo costo computacional e independencia respecto al motor de indexación. Por otra parte, [23, 24] aborda la siguiente cuestión: ¿cómo realizar un análisis morfológico sin recurrir a las nociones de morfema, afijo, exponente morfológico o cualquier representación de estos conceptos? Su modelo integra las propiedades semánticas y formales de las palabras de una manera uniforme y las representa en un grafo bipartito. Las caminatas aleatorias son utilizadas para simular la propagación de las activaciones en la red del léxico. El nivel de activación obtenido después de

[2] FLEMM está disponible en el sitio web: `http://www.univ-nancy2.fr/pers/namer/Telecharger_Flemm.htm`

[3] WINBRILL está disponible en el sitio web: `http://www.atilf.fr/scripts/mep.exe?HTML=mep_winbrill.txt;OUVRIR_MENU=1`

[4] TREETAGGER está disponible en el sitio web: `http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/DecisionTreeTagger.html`

la propagación indica la relación léxica de las palabras. Los miembros de la familia morfológica y la serie de derivación de una palabra se identifican entre sus vecinos léxicos por medio de analogías formales.

2.2. Aplicaciones

Aunque la constitución de familias morfológicas puede ser interesante en sí misma, su principal interés radica en los beneficios que produce su utilización como mecanismo de normalización (en lugar o como complemento del *stemming* o de la lematización) en ámbitos de aplicación concretos. Probablemente el dominio de aplicación más utilizado sea la indexación de términos en los sistemas de Recuperación de Información (RI). Efectivamente, en los últimos años se han publicado numerosos artículos que analizan la eficiencia, en diferentes idiomas, de las técnicas de *stemming*/lematización/familias en los sistemas de RI. Además, se han realizado avances importantes en RI en idiomas europeos diferentes del inglés. Las técnicas consideradas van desde algoritmos lingüísticos, como la normalización morfológica y la identificación de palabras compuestas, hasta técnicas sin usar conocimientos, como la indexación con n-gramas. En particular, [27] han realizado evaluaciones sobre los corpora de las campañas de evaluación de CLEF[5], que cubre ocho lenguas europeas. Sus resultados muestran que las técnicas de normalización morfológica aumentan la eficiencia de la RI y que pueden ser usadas de manera independiente del idioma. Además los algoritmos de disminución de variaciones sintácticas y morfosintácticas han demostrado una reducción importante de costos de almacenamiento y de gestión en RI [45] al almacenar lexemas en vez de términos.

[2] realizó un trabajo sobre los impactos de i) las plabras compuestas y ii) el *stemming* y la lematización en la RI monolingüe (inglés, finlandés, alemán y sueco) y bilingüe (lengua fuente inglés, lengua de traducción finlandés, alemán y sueco). En las pruebas monolingües, la búsqueda en un índice de palabras compuestas lemmatizadas da casi tan buenos resultados como la recuperación con un índice separado. Las diferencias se encuentran en el modo bilingüe: la búsqueda en un índice lematizado separado se comporta mejor que la búsqueda en un índice compuesto lematizado. Por otro lado, no se encontraron diferencias de rendimiento notables entre el *stemming* y lematización.

La lematización es eficaz, pero a menudo requiere de costosos recursos externos. El *stemming* es también eficaz en la mayoría de los contextos, normalmente casi tan bueno como la lematización y, por lo general, mucho menos costoso; además de que también tiene un efecto positivo en la ampliación de las *queries*. Sin embargo, en ambos casos, la idea es pasar de muchas formas flexivas de las palabras a un lema simple o *stem*, tanto en el índice de base de datos como en las consultas. Esto significa un esfuerzo adicional en la creación de índices de base de datos. [30] han adoptado un enfoque contrario, usando FCG (*Frequent Case (form) Generation*), lo cual consiste en dejar el índice de base de datos no normalizado y en enriquecer las consultas para cubrir las variaciones superficiales de las palabras clave. Pudiera pensarse que el procesamiento de las consultas con este enfoque es lento, pero los autores muestran que sólo importa procesar un número insignificante de las formas posibles de flexión, incluso en las lenguas morfológicamente complejas, para llegar a una eficiencia casi tan buena como la del *stemming* o de la lematización. [30] muestran también que, en muchas situaciones, sólo importa procesar los sustantivos y los adjetivos en las consultas. Ellos lo han implementado en alfabetos latino, griego y cirílico. Las aplicaciones incluyen, en particular, la RI en el Web en idiomas pobres en recursos morfológicos informatizados conocidos como lenguas π [6].

Sin embargo, la realidad es que los recursos lingüísticos necesarios para establecer las relaciones morfológicas sin necesidad de reglas pre-definidas no se encuentran disponibles para todos los idiomas y todos los dominios, sin hablar de la creación constante de neologismos [10]. Una posible solución puede ser la adquisición automática del conocimiento morfológico directamente a partir de los textos.

Nuestro trabajo propone una tercera vía, la lexematización, teniendo como objetivo el reagrupamiento morfológico de las palabras que pertenecen a una misma familia. Esto puede llevar accesoriamente a obtener una forma canónica de la familia, pero sin hacer uso del truncamiento como en el caso del *stemming*. Por lo tanto, lematizadores clásicos como TreeTagger o FreeLing[6] [3] llevan, por ejemplo, las palabras en inglés: *computing* a **computing**; *computation* y *computations* a **computation**; *computes* y *compute* a **compute**, *computer* y *computers* a **computer**. El *stemming* con algoritmos como el de Porter [38] o

[5] Cross-Language Evaluation Forum, `http://www.clef-campaign.org/`

[6] FreeLing está disponible en el sitio web: `http://www.lsi.upc.edu/~nlp/freeling/`

el de Paice [36] lleva esas variaciones a la forma truncada **comput**, que no es una palabra en inglés. Esto puede ser grave. El *stemming* lleva la lista *operate, operating, operates, operation, operative, operatives, operational* a **oper** que semánticamente no significa nada y que, peor aún, puede inducir a error y hacer perder precisión a los sistemas de extracción de información que trabajan a base de *queries*. En el caso del español, el lematizador de FreeLing lleva *computación* a **computación**; *computadora* y *computadoras* a **computador**; *cómputos* a **cómputo**; *computó, computar, computamos, computaron* a **computar**. En francés el analizador Flemm tiene un comportamiento similar, pues lleva **computabilités** a *computable*, **computationnels**, **computationnel** y **computationnelle** a *computation*. Por el contrario, el reagrupamiento morfológico en familias de palabras que proponemos podría llevar, en la mayor parte de los casos, todas esas variaciones a una forma única lexematizada y semánticamente significativa: **compute**, por ejemplo en inglés, **computar**, en español, o **computationnel**, en francés.

Hay dos puntos fundamentales que diferencian a nuestro sistema de las salidas de los algoritmos de la literatura: primero, el lexema es cognitivamente más intuitivo que un *stemm* truncado y, segundo y más importante, el grado de reagrupamiento de la lexematización es superior al de la lematización o del *stemming* porque produce una forma única. La obtención de una forma única para una familia tiene grandes ventajas para los sistemas de extracción y de recuperación de información, al disminuir la dimensión del espacio de representación utilizado [4, 34].

3. Matriz de probabilidades (sufijos × prefijos)

En este primer algoritmo *naïf* utilizamos un modelo probabilístico simple a través de una matriz que maximiza el producto de (sufijos × prefijos), condicionada por la probabilidad de los n-gramas de letras, con $n = 1, 2, ..., 6$. Asociamos a cada sufijo y a cada prefijo una probabilidad estimada por el número de ocurrencias de sufijos y prefijos, dividido por el número de ocurrencias total.

Sea $\{\text{suf}_1, \text{suf}_2, ..., \text{suf}_m\}$ la lista de m sufijos. Sea $\mathcal{N}(\text{suf}_i)$ el número de ocurrencias de cada sufijo i. La probabilidad del sufijo suf_i será estimada por:

$$p(\text{suf}_i) = \frac{\mathcal{N}(\text{suf}_i)}{\sum \mathcal{N}(\text{suf}_i)} \tag{1}$$

De la misma manera, sea $\{\text{pref}_1, \text{pref}_2, ..., \text{pref}_n\}$ la lista de n prefijos. Sea $\mathcal{N}(\text{pref}_j)$ el número de ocurrencias de cada prefijo j. La probabilidad del prefijo pref_j será estimada por:

$$p(\text{pref}_j) = \frac{\mathcal{N}(\text{pref}_j)}{\sum \mathcal{N}(\text{pref}_j)} \tag{2}$$

La matriz $\alpha_{[n\times m]}$, que combina la información de prefijos y sufijos de una palabra, estará definida por:

$$\alpha = \begin{pmatrix} \alpha_{11} & \alpha_{12} & \cdots & \alpha_{1j} & \cdots & \alpha_{1m} \\ \alpha_{21} & \alpha_{22} & \cdots & & \cdots & \alpha_{2m} \\ \vdots & \vdots & \ddots & \vdots & & \vdots \\ \alpha_{i1} & \alpha_{i2} & \cdots & \alpha_{ij} & \cdots & \alpha_{im} \\ \vdots & \vdots & & \vdots & \ddots & \vdots \\ \alpha_{n1} & \alpha_{n2} & \cdots & \alpha_{nj} & \cdots & \alpha_{nm} \end{pmatrix} \tag{3}$$

donde cada elemento α_{ij} de esta matriz es el producto de las probabilidades del prefijo j (columna) con el sufijo i (renglón): $\alpha_{ij} = p(\text{suf}_i) \times p(\text{pref}_j)$. Con propósitos de introducir nuestro algoritmo, hemos definido las hipótesis razonables siguientes:

- **Hipótesis I.** La raíz de una palabra está définida por el n-grama que corresponde a la probabilidad máxima de Max$\{\alpha_{ij}\}$.
- **Hipótesis II.** Existe la posibilidad de ausencia del prefijo o del sufijo, es decir, una cadena vacía puede ser sufijo o prefijo.

Por ejemplo, para calcular el lexema de la palabra francesa *représentations*[7], procedemos como sigue: las probabilidades, según un corpus hipotético, de los prefijos (con $n \leq 5$ caracteres) son: $p(\mathbf{r})$=0,23, $p(\mathbf{re})$=0,42, $p(\mathbf{rep})$=$p(\mathbf{repr})$=0,01, $p(\mathbf{repré})$=0,001; y de los sufijos: $p(\mathbf{tions})$=0,02, $p(\mathbf{ions})$=0,01, $p(\mathbf{ons})$ =0,05, $p(\mathbf{ns})$=0,2, $p(\mathbf{s})$=0,78. De esta forma, el valor $\text{MAX}\{p(\text{suf}_i)\times p(\text{pref}_j)\}$ corresponde al par {**re, s**}, es decir $\alpha_{\mathbf{re,s}} = 0{,}42 \times 0{,}78$, y el lexema calculado será **présentation**[8]. Otros ejemplos permiten ilustrar el uso de los sufijos vacíos: supongamos que **er** y **ons** son sufijos válidos, **@** es la subcadena vacía y todos ellos fueron detectados por el algoritmo de sufijos (según la hipótesis I). Sea la lista con las siguientes palabras en francés a lexematizar: $\mathcal{L}$ = {*réaliser, réalise, parlons, parler, présenter, représentation, représentations*}. El lexema de la palabra **réaliser** será **réalise** porque: i) la palabra **réaliser = @ + réalis + er**, ii) además **@** es un prefijo y **er** un sufijo, ambos válidos. Por otro lado, el lexema de **parlons** según este algoritmo será **parlons**, porque incluso si **ons** resulta ser un sufijo válido en la lista, **parl** no existe como palabra en $\mathcal{L}$.

Sin embargo este método presenta limitaciones al nivel de la extracción de raíces. Puesto que la estimación de los candidatos n-gramas como lexemas utiliza la probabilidad máxima (producto de las probabilidades de los sufijos y prefijos), la subcadena vacía es considerada como un prefijo con una probabilidad más grande que la de los otros prefijos detectados. Pero si se hace caso omiso de la cadena vacía se impone la búsqueda de prefijos, sabiendo que hay palabras que no contienen prefijos. Por otra parte, no se pueden validar todos los n-gramas detectados como lexemas salvo si se presentan como palabras existentes en el corpus. Por tanto, decidimos proponer otro método que combina una técnica vectorial con la firma [7] de los sufijos para disminuir estas limitaciones. Este será el objeto de la sección siguiente.

4. Algoritmo vectorial y cálculo de prefijos

El método siguiente se basa en un mecanismo de reagrupación de familias. Después de una etapa de pre-procesamiento (eliminación de símbolos de puntuación y de normalización de mayúsculas) del texto, la lista que contiene las j palabras tipo extraídas es transformada en una representación vectorial para crear una matriz $\mathcal{M}_{[i\times j]}$ de j columnas (las palabras) e i filas que contienen los n-gramas prefijos. Esta matriz se utiliza para detectar los vectores similares. El algoritmo se desarrolla en dos etapas:

1. Cálculo vectorial de prefijos.
2. Corrección por la firma de la familia.

La matriz $\mathcal{M}$ será construída iterativamente en un cierto número de ciclos, que será fijado a cinco por razones que serán explicadas más adelante. En efecto, cada iteración corresponde a una longitud fija de n-gramas prefijos. Por lo tanto, el objetivo de las iteraciones es eliminar las palabras que ya están asignadas a F, es decir, reagrupadas. Esto evita el riesgo de aumentar los errores de reagrupamiento de palabras que podrían ser consideradas de la misma familia sin serlo en realidad. El conjunto F de las familias iniciales debe ser corregido con un algoritmo de sufijación. Este algoritmo se basa en los conceptos de la firma e independencia entre las familias. Por último, la lista de las familias corregida se valida una vez más calculando su grado de similitud.

4.1. Matriz de similitud

El algoritmo toma como entrada una lista $\mathcal{L}$ de m palabras $\{w_1, ..., w_j, ..., w_m\}$. El proceso de reagrupamiento construye la matriz de similitud $\mathcal{M}_{[i\times j]}$ con la lista de palabras como columnas y como líneas n prefijos constituídos por n-gramas de caracteres. A cada palabra w_j se le asigna un valor binario 0 ó 1. El valor 1 significa que la palabra w_j comienza con el n-grama i y 0 indica el caso contrario. Inicialmente, existen tantas familias como palabras m. El algoritmo reagrupa en familias las palabras que representan vectores similares. Para minimizar el riesgo de reagrupar palabras morfológicamente alejadas pero que tienen los mismos n-gramas prefijos, el proceso será iterativo. En cada iteración se disminuirá la longitud del n-grama en cuestión. Incialmente la longitud es $l = 8$ y el objetivo será eliminar en cada

[7] Representaciones.
[8] Presentación.

iteración, es decir $l = [8, 7, 6, 5, 4]$, las palabras ya reagrupadas para disminuir la introducción de errores de reagrupamiento.

Para ilustrar el funcionamiento del algoritmo, presentamos el siguiente ejemplo de palabras en inglés. Sea la lista $\mathcal{L}$, conteniendo las $m = 18$ palabras a reagrupar:

$\mathcal{L} = \{w_1$: **involving**, w_2: **investigating**, w_3: **internationally**, w_4: **development**, w_5: **expressions**, w_6: **involved**, w_7: **investigator**, w_8: **investigations**, w_9: **developments**, w_{10}: **expression**, w_{11}: **interfering**, w_{12}: **integration**, w_{13}: **international**, w_{14}: **interior**, w_{15}: **interim**, w_{16}: **director**, w_{17}: **dir**, w_{18}: **directive**$\}$

Las w_j palabras, $j = 1, \ldots, 18$ representan las columnas de la matriz de similitud. De cada palabra w_j se extraen los n-grammas de caracteres prefijos. El problema que se presenta es cómo definir inteligentemente la longitud n de los n-gramas. Empíricamente decidimos realizar cinco iteraciones, de manera similar a la Hipótesis **I** que maximiza las probabilidades, puesto que el uso de cinco letras es al mismo tiempo un buen compromiso para detectar los sufijos. En cada iteración se eliminará un cierto número k de palabras que han sido probablemente reagrupadas. De esta manera se obtendrán cinco matrices con grupos de n-gramas y palabras diferentes:

- Iteración 1: $\mathcal{L}^{(1)}$ con 18 palabras $j = 1, 2, \ldots, 18$; once 8-gramas: ng_i = {**involvin, investig, internat, developm, expressi, involved, interfer, integrat, interior, director, directiv**}.

Cuadro 1: Matriz de similitud $\mathcal{M}_{[11\times18]}$, iteración 1

	$w_{j=1}$	**2**	**3**	**4**	**5**	6	**7**	**8**	**9**	**10**	11	12	**13**	14	15	16	17	18
$ng_{i=1}$(involvin)	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2(investig)	0	**1**	0	0	0	0	**1**	**1**	0	0	0	0	0	0	0	0	0	0
3(internat)	0	0	**1**	0	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0
4(developm)	0	0	0	**1**	0	0	0	0	**1**	0	0	0	0	0	0	0	0	0
5(expressi)	0	0	0	0	**1**	0	0	0	0	**1**	0	0	0	0	0	0	0	0
6(involved)	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
7(interfer)	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
8(integrat)	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
9(interior)	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
10(director)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
11(directiv)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Esta operación reagrupa $k = 9$ palabras $\{w_2$: investigating, w_7: investigator, w_8: investigations$\}$, $\{w_5$: expressions, w_{10}: expression$\}$, $\{w_3$: internationally, w_{13}: international$\}$ y $\{w_4$: development, w_9: developments$\}$. Estas 9 palabras (índices en fondo gris y negritas) se eliminarán de la lista $\mathcal{L}$.

- Iteración 2: $\mathcal{L}^{(2)}$ con nueve palabras $j \neq (2, 3, 4, 5, 7, 8, 9, 10, 13)$; ocho 7-gramas: ng_i= {**involvi, involve, interfe, integra, interio, interim, directo, directi**}.

Cuadro 2: Matriz de similitud $\mathcal{M}_{[8\times9]}$, iteración 2

	$w_{j=1}$	6	11	12	14	15	16	17	18
$ng_{i=1}$(involvi)	1	0	0	0	0	0	0	0	0
2(involve)	0	1	0	0	0	0	0	0	0
3(interfe)	0	0	1	0	0	0	0	0	0
4(integra)	0	0	0	1	0	0	0	0	0
5(interio)	0	0	0	0	1	0	0	0	0
6(interim)	0	0	0	0	0	1	0	0	0
7(directo)	0	0	0	0	0	0	1	0	0
8(directi)	0	0	0	0	0	0	0	0	1

En la iteración 2 ninguna familia fue detectada.

- Iteración 3: $\mathcal{L}^{(3)}$ con nueve palabras $j \neq (2,3,4,5,7,8,9,10,13)$; cinco 6-gramas: ng_i=**{involv, interf, integr, interi, direct}**.

Cuadro 3: Matriz de similitud $\mathcal{M}_{[5\times9]}$, iteración 3

	$w_{j=}$**1**	**6**	11	12	**14**	**15**	**16**	17	**18**
$ng_{i=}$1(involv)	**1**	**1**	0	0	0	0	0	0	0
2(interf)	0	0	1	0	0	0	0	0	0
3(integr)	0	0	0	1	0	0	0	0	0
4(interi)	0	0	0	0	**1**	**1**	0	0	0
5(direct)	0	0	0	0	0	0	**1**	0	**1**

Esta iteración reagrupa $k = 6$ palabras {w_1: involving, w_6: involved}, {w_{14}: interior, w_{15}: interim} y {w_{16}: director, w_{18}: directive}, que serán eliminadas de la lista a procesar (cuadro 3).

- Iteración 4: $\mathcal{L}^{(4)}$ con tres palabras $j \neq$ (1,2,3,4,5,6,7,8,9,10,13,14,15,16,18); dos 5-gramas: ng_i=**{inter, integ}**.

Cuadro 4: Matriz de similitud $\mathcal{M}_{[2\times3]}$, iteración 4

	$w_{j=}$11	12	17
$ng_{i=}$1(inter)	1	0	0
2(integ)	0	1	0

En la iteración 4 ninguna familia fue detectada (cuadro 4).

- Iteración 5: $\mathcal{L}^{(5)}$ con tres palabras $j \neq$ (1,2,3,4,5,6,7,8,9,10,13,14,15,16,18); un 4-grama: ng_1=**inte**.

Cuadro 5: Matriz de similitud $\mathcal{M}_{[1\times3]}$, iteración 5

	$w_{j=}$**11**	**12**	17
$ng_{i=}$1(inte)	**1**	**1**	0

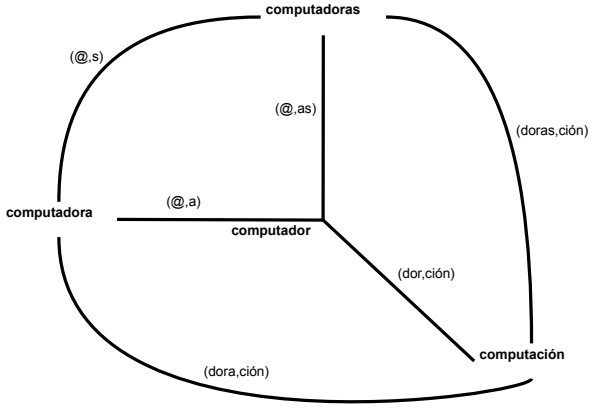
La iteración final reagrupa las palabras {w_{11}: interfering, w_{12}: integration} y el algoritmo se detiene (cuadro 5).

El cuadro 6 presenta las palabras reagrupadas en 9 familias, en función de las iteraciones sucesivas. La familia número 9 queda como una familia mono-palabra con el elemento w_{17}:**dir**.

Cuadro 6: Conjunto de familias F

Iteración	N° de familia	w$_j$	Palabras de la misma familia
1	1	w$_5$, w$_{10}$	expressions, expression
1	2	w$_3$, w$_{13}$	internationally, international
1	3	w$_4$, w$_9$	development, developments
1	4	w$_2$, w$_7$, w$_8$	investigating, investigator, investigations
3	5	w$_1$, w$_6$	involved, involving
3	6	w$_{14}$, w$_{15}$	interior, interim
3	7	w$_{16}$, w$_{18}$	director, directive
5	8	w$_{11}$, w$_{12}$	interfering, integration
5	9	w$_{17}$	dir

Figura 1: Identificación de la firma



4.2. Sufijación mediante la firma

Las familias reagrupadas pueden contener errores. En particular, en nuestro ejemplo las siguientes familias {w_{14}: interior, w_{15}: interim}, {w_{11}: interfering, w_{12}: integration} y {w_{17}:dir} no están correctamente reagrupadas y deberían corregirse. Para ello se necesita utilizar la información contenida en los sufijos. Después de la formación de familias independientes (palabras reagrupadas de manera exclusiva), extraemos la subcadena común más larga, de las palabras pertenecientes a la familia. Para la sustitución de estas palabras con relación a sus subcadenas comunes detectadas, se obtiene una lista de cadenas deducidas que serán consideradas como sufijos iniciales de n caracteres. El par de sufijos que contengan un número de ocurrencias superior o igual a dos serán llamados *componentes de la firma de la familia*. Para comprobar si un par de sufijos son componentes de una firma, sólo será necesario comprobar las dos condiciones siguientes:

1. (Sufijo$_1$ $\in$ familia F_i, Sufijo$_2$ $\in$ familia F_j) $\Longrightarrow F_i \neq F_j$
2. $\mathcal{N}$(Sufijo$_1$) && $\mathcal{N}$(Sufijo$_2$) ≥ 2

En el caso afirmativo de las dos condiciones, Sufijo$_1$ y Sufijo$_2$ serán los componentes de la firma. La primera condición puede tener ambigüedad en el caso que los dos sufijos pertenezcan a familias diferentes. Sin embargo, el último caracter de la cadena común elimina esta ambigüedad. El principal interés de la firma es extraer sus componentes para convertirlos en sufijos más robustos. Estos se utilizarán para mejorar el agrupamiento previo de palabras morfológicamente similares.

Por ejemplo, sea la familia obtenida a través de los cálculos de la matriz de similitud:

$\mathcal{F}$ = { computador, computadoras, computadora, computación };

Los siguientes seis pares de sufijos (ver la figura 1) serán detectados:

Sufijos: { (**@**,s), (**@**,a), (**@**,as), (dor, ción), (dora, ción), (doras, ción) }

El símbolo **@** significa sufijo vacío. Los ocho componentes de la firma (con sus frecuencias entre paréntesis) son: **@** (3), **ción** (3), **doras** (1), **dora** (1), **dor** (1), **as** (1), **s** (1) y **a** (1). La firma de esta familia será **(@, ción)**.

El cuadro 7 muestra los sufijos encontrados, los candidatos y la firmas de la familia correspondiente del ejemplo presentado en sección 4.1. La subcadena **dir**, con el eventual sufijo **@**, no es considerada pues su frecuencia es inferior a dos.

Debe notarse que las familias 1 y 3 poseen la misma firma, según se muestra en la columna "Candidato a firma" del cuadro 7. Es necesario un análisis más fino para distinguirlas. El último caracter de la subcadena correspondiente permitirá separarlas correctamente: **n** para la familia 1, **t** para la 3, lo que genera las firmas 1:**(n@, s)** y 3:**(t@, s)**.

Cuadro 7: Firma de las familias

Nº de familia	Subcadena	Sufijos	Candidato a firma	Firma de cada familia
1	**expression**	@, s	(@, s)	**(n@, s)**
2	**international**	ly, @	(@, ly)	**(@, ly)**
3	**development**	@, s	(@, s)	**(t@, s)**
4	**investigat**	ing, or, ions	(ing, or)	**(ing, or)**
5	**involv**	ing, ed	(ing, ed)	**(ing, ed)**
6	**interi**	or, m	(or, m)	**(or, m)**
7	**direct**	or, ive	(or, ive)	**(or, ive)**
8	**inte**	rfering, gration	- -	

Corrección de las primeras familias mediante la firma

La combinación de los resultados de las primeras familias F con los componentes de la firma permite mejorar el rendimiento del reagrupamiento y de eliminar una serie de palabras inadecuadas en las familias en las cuales fueron destinadas. Esta etapa también permite detectar nuevos sufijos utilizando una longitud fija que debe tomarse en cuenta cuando la palabra cambia de una familia inicial a otra. La familia 8 será corregida y separada en dos (familia 8, *interfering* y familia 8', *integration*), porque los sufijos **{rfering, gration}** no son componentes de las firmas y además su longitud no permite incluirlos en la base de sufijos verdaderos.

Reagrupamiento final

Se aplicó nuevamente el método vectorial pero con una sola iteración, porque la lista de los n-gramas prefijos es conocida (independientemente de la longitud de los n-gramas). Esta etapa permite reagrupar las subfamilias gracias a los n-gramas prefijos que son deducidos de los últimos sufijos (robustos).

Finalmente, el lexema de las palabras se detecta usando la familia que las reagrupa, mediante los dos criterios siguientes:

- La palabra con la menor longitud de caracteres será considerada como el lexema de la familia.
- En el caso de dos palabras con igual longitud, el lexema será seleccionado de acuerdo a la palabra que tenga mayor número de ocurrencias de sus sufijos.

El cuadro 8 muestra en la parte de arriba, las palabras en inglés del ejemplo de la sección 4.1, reagrupadas en siete familias (las familias mono-palabras 8 y 8' no son consideradas) después de la corrección por la firma. El mismo cuadro muestra en el medio algunos ejemplos en francés, donde se observa que el reagrupamiento morfológico es diferente de la lematización. Por ejemplo, en francés el lexema calculado de la palabra *passer* fue **passe** y no **passer**[9], que sería lo correcto. Esto se justifica por el criterio de la longitud mínima de la palabra, que fue considerada en la familia que la reagrupó. En la parte baja del cuadro 8, se presentan algunos ejemplos en español, sus reagrupamientos y los lexemas obtenidos. Una vez más es patente que la lexematización lleva las formas verbales (**utilización, utilizados, utilizando, utilizan, utilizarse, utilizar, utilizará, utiliza**) a **utiliza** y no al lema **utilizar**, que sería lo correcto, pero la restricción de la palabra más corta con mayor frecuencia, se impone nuevamente.

Otro caso interesante es la familia número 6, en inglés, donde el lexema **interim** representa las palabras *interior* e *interim*, lo cual es evidentemente un error que el algoritmo de sufijos es incapaz de corregir. Un nivel más fino de análisis sería necesario para paliar este tipo de limitaciones.

5. Evaluación

La comparación directa de nuestro algoritmo con los resultados de otros sistemas no resulta ser fácil dada la variabilidad de los objetivos a alcanzar en cada uno de ellos. Por ejemplo, el sistema FLEMM

[9]Respectivamente **pasa** y **pasar**.

Cuadro 8: Familias finales

Lista de palabras	Lexema
INGLÉS	
expression, expressions	**expression**
internationally, international	**international**
development, developments	**development**
investigating, investigator, investigations	**investigator**
involving, involved	**involved**
interior, interim	**interim**
director, directive	**director**
FRANCÉS	
demanderais, demandait, demander, demandons	**demander**
universe, universes, universelle	**universe**
particulière, particulièrement, particulier, particulières	**particulier**
refusera, refuse, refuser	**refuse**
passer, passe, passons, passions	**passe**
ESPAÑOL	
añada, añade, añadió, añaden, añades	**añade**
abandonadas, abandonados, abandonan, abandonar	**abandonar**
absolutamente, absoluto, absolutos, absolutismo	**absoluto**
utilización, utilizados, utilizando, utilizan, utiliza, utilizarse, utilizar, utilizará	**utiliza**
votación, votaciones, votaciones, votaremos, votarán, votar, votará	**votar**

obtiene lemas y no lexemas. SNOWBALL[10] obtiene raíces o *stems* truncados. TREETAGGER o FREELING tampoco reagrupan familias, lo que hace difícil comparar la precisión. Por el contrario, nuestro método reagrupa las familias de palabras y después calcula su lexema, independientemente del idioma y del contexto. Para evaluar nuestro sistema hemos utilizado corpus en tres idiomas: inglés, francés y español. Estos corpus y sus *gold standard* manuales han sido construidos y están puestos a disposición de la comunidad científica[11]. Las listas de las palabras a reagrupar contienen entre 1 000 y 5 000 formas tipo diferentes.

Establecimos un protocolo de evaluación manual con una longitud fija (longitud de cálculo de los sufijos), limitada a 6 caracteres. Calculamos la precisión de las familias P_f y la precisión de las palabras P_w reagrupadas encontradas en sus familias correctas. Todas las palabras de entrada fueron utilizadas en el protocolo de evaluación. Así, hemos calculado:

$$P_w = \frac{\text{Palabras correctamente reagrupadas}}{\text{Total de palabras reagrupadas}} \tag{4}$$

[10] SNOWBALL está disponible en el sitio web: `http://snowball.tartarus.org/`

[11] Otros corpus, como los de CLEF, son accesibles únicamente a los miembros participantes del consorcio. Los corpus usados en este artículo pueden ser recuperados en el sitio Web `http://www.laboratorio.pais`

$$P_f = \frac{\text{Familias correctamente reagrupadas}}{\text{Total de familias}} \quad (5)$$

El cuadro 9 muestra los resultados obtenidos de acuerdo con el protocolo de evaluación. El español presenta el mayor número de inflexiones y de variabilidad, seguido del francés y finalmente del inglés. Nuestros resultados de reagrupamiento y lexematización reflejan esta complejidad. Efectivamente, el inglés obtiene el valor más alto, de **96,8** % en la precisión de las palabras correctamente reagrupadas, seguido del francés y del español. En cuanto a la precisión sobre las familias, el francés obtiene el primer lugar con **96,4** %, seguido del inglés y en tercer lugar del español.

Cuadro 9: Resultados

Corpus	Nº Total de familias	Familias correctamente reagrupadas	P_f Precisión de familias %	Palabras correctamente reagrupadas	Nº Total de palabras reagrupadas	P_w Precisión de palabras reagrupadas %
Inglés	893	843	94,40	2 082	2 150	**96,84**
Francés	1 414	1 363	**96,39**	3 753	3 917	95,81
Español	1 369	1 277	93,28	2 828	3 119	90,67

Nuestro método confirma su capacidad para resolver el problema de elección de la longitud para reagrupar las palabras en la etapa inicial. La sufijación interviene únicamente en el reagrupamiento de los vectores de palabras para construir una base sólida de sufijos. Este método tiene la ventaja de ser independiente a la sufijación después de la etapa inicial para reagrupar las palabras de una misma familia. Además, se realiza el cálculo de sufijos que son difíciles de detectar, cuando una base sólida de sufijos no existe. Lo ingenioso de esta regla es que respeta la prioridad de los sufijos reales ya detectados y la prioridad del reagrupamiento inicial. Sin embargo, una de las desventajas es la complejidad del algoritmo (tamaño de la matriz).

Aún siendo difícil una comparación directa con otros métodos, hemos encontrado que [7] reporta una precisión de entre 91,5 y 93,1 para listas de palabras evaluadas manualmente en francés, y de entre 89,8 y 91,2 para el inglés, con un método ascendente de reagrupamiento jerárquico. Nuestro algoritmo, capaz de procesar listas independientemente del idioma, está mejor posicionado (95,8 y 96,8 respectivamente en francés e inglés), comparable al algoritmo de la distancia de Levenshtein presentado en [14] (que logra un 96 % de precisión en inglés). Estudios como el de [35] muestran que el regrupamiento de variantes morfológicas, similar al nuestro en el sentido de no necesitar ni conocimientos *a priori* de la lengua ni recursos externos, presenta un interés superior al del *stemming* o al de la lematización, en tareas específicas de RI como el *query expansion*. Finalmente, hemos realizado un experimento piloto sobre un pequeño corpus en ruso pero sin el módulo de sufijos. El resultado para la precisión de familias fue de aproximadamente 58 %, lo que confirma que la técnica de sufijos es una estrategia interesante que mejora realmente la precisión del reagrupamiento.

6. Discusión, conclusión y perspectivas

La principal limitación de las herramientas existentes en lematización o *stemming* es que están, en la mayoría de los casos, basadas en recursos externos, tales como listas de afijos, reglas morfológicas o diccionarios. En este artículo hemos presentado un método no supervisado para obtener las familias morfológicas y la lexematización de palabras sin utilizar su contexto. Los resultados en el corpus de test obtenidos por nuestro método fueron calculados de manera exhaustiva sobre el conjunto completo de datos disponibles. Es decir, no hubo validación cruzada ni fueron necesarios conjuntos de entrenamiento o de validación porque este método no requiere de ningún aprendizaje previo. Tampoco se requiere el ajuste de parámetros internos: el sistema afina en cada iteración el agrupamiento de la familia morfológica. Esto representa una ventaja adicional sobre los algoritmos clásicos de aprendizaje supervisado, como los *Support Vector Machines* (SVM) [43, 44] o las redes de neuronas artificiales [26], que necesitan de (grandes) conjuntos de aprendizaje previamente etiquetados. Lo mismo puede decirse de los algoritmos

probabilísticos, que en general necesitan que el conjunto de aprendizaje sea de tamaño suficiente para poder detectar las características (*features*) adecuadas para la tarea.

En todo momento buscamos crear un algoritmo estadístico simple. Esto lo hace especialmente atractivo para ser acoplado a sistemas RI de resúmen automático[12]. El algoritmo utiliza iterativamente la información morfológica para construir las familias. Esto evita usar las proyecciones de tipo *kernel* hacia espacios de mayor dimensionalidad (tal como lo hacen las SVM) como función de similitud. Algoritmos SVM de reagrupamiento como el presentado en [5] son interesantes, pero requieren la definición de algunos parámetros para funcionar. En otros casos, los métodos de reagrupamiento clásicos semi-supervisados (que requieren de un conjunto de aprendizaje mínimo) o no supervisados pueden ser dependientes de la naturaleza de los datos o necesitar que les sea indicado precisamente el número de grupos esperados. Nada de esto es necesario en nuestro caso.

La obtención de lexemas presenta un interés superior a las formas truncadas producidas por los algoritmos de *stemming* [28, 33, 36, 41] a nivel semántico, y aumentan la precisión de los *queries* de los sistemas de extracción de información (se obtienen factores de compresión de hasta 50 % en los sistemas de RI almacenando lexemas en lugar de los términos) [35]. Nuestro algoritmo es un método sencillo, con resultados razonablemente buenos, especialmente en términos de precisión de palabras reagrupadas. Por otra parte, el reagrupamiento y la lexematización se realizan solamente a partir de una lista de palabras de entrada y no utilizan ningún tipo de recursos externos (ni lingüísticos ni bases de conocimiento), a diferencia de métodos clásicos para la obtención de raíces basados en reglas. Estos modelos utilizan reglas para las excepciones y requieren recursos que deben ser adaptados a otras lenguas, que significa un trabajo difícil y tedioso.

Sin embargo, debemos decir que existen grandes diferencias entre lenguas con flexión pobre (como por ejemplo, el inglés) y lenguas con flexión rica (por exemplo, el finlandés), por no hablar de las lenguas aglutinantes, como el vasco o el japonés. La riqueza morfológica de estas últimas lenguas es superior al de las tres lenguas utilizadas en este trabajo (inglés, francés y español). Deberán realizarse investigaciones más profundas para mostrar la validez de los métodos expuestos en este trabajo. Por ejemplo, en particular ¿el sistema podría aplicarse al árabe? Es difícil responder claramente a esta problématica, porque la lengua árabe es altamente derivativa: las palabras derivan de raíces formadas por 3, 4 o rara vez superior a 5 vocales. Además es difícil separar morfología flexiva de derivacional. Esto merece una reflexión sobre la eficiencia de la tabla de n-gramas de nuestro método vectorial aplicado al árabe, porque en las lenguas evaluadas en este artículo, esta tabla se basa únicamente en los n-gramas de prefijos. Un trabajo actualmente en curso[13] incluye la lexematización de la lengua somalí[14]. Esta lengua aglutinante, y con pocos recursos computacionales [1], usa prefijos y sufijos, pero de una manera mucho más rígida y dentro de la misma palabra (temporalidad independiente de la flexión). Sin embargo la mayoría de los verbos son regulares, lo que en principio debería facilitar la tarea de lexematización.

En este artículo hemos respondido también al problema que describe [7] con relación a la capacidad de autonomía para producir un reagrupamiento adecuado utilizando unicamente los sufijos. El reagrupamiento de las familias con este método no utiliza ni prefijos ni sufijos para empezar las iteraciones y tampoco necesita detectar el idioma.

Las palabras que presentan problemas a los lematizadores clásicos (como pueden ser los sustantivos, las palabras en lengua extranjera o las palabras desconocidas (OOV)) no representan mayor dificultad a nuestro método, siempre y cuando su frecuencia permita un procesamiento estadístico apropiado. Los resultados han confirmado la independencia de reglas predefinidas y de la lengua. Otros experimentos, con una longitud de más de cinco caracteres, confirman la tendencia mostrada en el cuadro 9, con precisiones ligeramente inferiores de palabras y familias correctamente reagrupadas. Las perspectivas para mejorar el sistema son varias. Entre otras, el acoplamiento con un sistema más robusto de detección de sufijos, como por ejemplo el descrito por [42]. Otra idea es la de utilizar la firma de las familias para controlar la eventual fusión de varias familias en una sola. También queremos aumentar las pruebas para verificar la independencia de la lengua de nuestro método. Para ello pensamos usar los corpora de CLEF, en ocho

[12] Algunas pruebas están llevándose a cabo con sistemas de resuúmen automático de documentos como Cortex [8].

[13] Colaboración en curso con el *Institut de Sciences et Nouvelles Technologies du Centre d'Etudes et de Recherche de Djibouti* (CERD), site Web `http://www.cerd.dj.`

[14] La lengua somalí es miembro de la rama oriental de las lenguas cusitas (familia afroasiática). Se habla sobre todo en África del Este: Somalia, Yibutí, Etiopía, y Kenia. El número exacto de hablantes es desconocido, pero se estima de entre 15 y 25 millones de personas. Es una lengua π, poco dotada de recursos informatizados.

idiomas europeos. A pesar de sus ventajas, el método tiene el defecto de necesitar varias (menor o igual que 5) iteraciones por palabra para converger. Sin embargo, en listas provenientes de textos de tamaño razonable (como es el caso típico de los sistemas de extracción de información tales los resumidores automáticos de documentos) esto no representa mayor problema.

Agradecimentos

El autor agradece los consejos y las correcciones de los árbitros anónimos, los pertinentes comentarios de Gerardo Sierrra, del Grupo de Ingeniería Lingüistíca (GIL, Instituto de Ingeniería), Universidad Nacional Autónoma de México, Iria Da Cunha, GIL/UNAM e IULA (Barcelona) y de Patricia Velázquez (VM Labs Avignon), que contribuyeron a aumentar la claridad y pertinencia de este artículo. Se agradecen también algunas de las pruebas del algoritmo realizadas por Nabi Rachid y Jalal Bouhafer, así como las discusiones e intercambio de ideas con Silvia Fernández.

Referencias

[1] A. Nimaan and P. Nocera and Torres-Moreno, J-M. Boîtes à outils TAL pour les langues peu informatisées: le cas du Somali. In *Journées d'Analyses des Données Textuelles (JADT'06)*, pages 697–705, Besançon-France, 2006.

[2] E. Airio. Word normalization and decompounding in mono- and bilingual IR. *Information Retrieval*, 9(3):249–271, 2006.

[3] J. Atserias, B. Casas, E. Comelles, M. González, L. Padró, and M. Padró. FreeLing 1.3: Syntactic and semantic services in an open-source NLP library. In *fifth International Conference on Language Resources and Evaluation (LREC'06), ELRA*, 2006.

[4] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison Wesley, 1999.

[5] A. Ben-Hur, D. Horn, H.T. Siegelmann, and V. Vapnik. Support Vector Clustering. *Journal of Machine Learning Research*, 2:125–137, 2001.

[6] V. Berment. *Méthodes pour informatiser des langues et des groupes de langues peu dotées*. PhD thesis, Joseph Fourier Univ. Grenoble, 2004.

[7] D. Bernhard. Apprentissage non supervisé de familles morphologiques par classification ascendante hiérarchique. In *TALN'07*, volume 1, pages 367–376, 2006.

[8] F. Boudin and J.-M. Torres-Moreno. NEO-CORTEX: A Performant User-Oriented Multi-Document Summarization System. In *Computational Linguistics and Intelligent Text Processing (CICLing'07)*, volume 4394 of *Lecture Notes in Computer Science*, pages 551–562. Springer, 2007.

[9] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Wadsworth and Brooks, Monterey, CA, 1984.

[10] M.T Cabré Castellví. Typology of neologisms: a complex task. *Alfa (São Paulo)*, 50(2):229–250, 2006.

[11] M. Creutz and K. Lagus. Unsupervised Discovery of Morphemes. In *6th Workshop of the ACL Special Interest Group in Computational Phonology (SIGPHON)*, pages 21–30, 2002.

[12] M. Creutz and K. Lagus. Unsupervised morpheme segmentation and morphology induction from text corpora using Morfessor 1.0. Technical Report A81, Publications in Computer and Information Science, Helsinki University of Technology, 2005.

[13] R. Gómez Díaz. *Lematización en español: una aplicación para la recuperación de información*. Agapea Books, Spain, 2005.

[14] D.P. Lyras and K.N. Sgarbas and N.D. Fakotakis. Using the Levenshtein Edit Distance for Automatic Lemmatization: A Case Study for Modern Greek and English. In *19th IEEE International Conference on Tools with Artificial Intelligence - (ICTAI'07)*, volume 2, pages 428–435, 2007.

[15] F. Namer. Flemm: Un analyseur Flexionnel de Français à base de règles. In Christian Jacquemin, editor, *Traitement automatique des Langues pour la recherche d'information*, pages 523–547. Hermes, 2000.

[16] C.G. Figuerola, R. Gómez Díaz, and E. López de San Román. Stemming and n-grams in Spanish: An evaluation of their impact on information retrieval. *Journal of Information Science*, 26(6):461–467, 2000.

[17] A.F. Gelbukh, M. Alexandrov, and S.-Y. Han. Detecting inflection patterns in natural language by minimization of morphological model. In Sanfeliu A., Trinidad J.F.M., and Carrasco-Ochoa J.A., editors, *9th Iberoamerican Congress on Pattern Recognition (CIARP'04), Progress in Pattern Recognition, Image Analysis and Applications*, volume 3287, pages 432–438. Lecture Notes in Computer Science, Springer-Verlag, Berlin, 2004.

[18] A.F. Gelbukh and G. Sidorov. Approach to construction of automatic morphological analysis systems for inflective languages with little effort. In *Computational Linguistics and Intelligent Text Processing (CICLing'03)*, volume 2, pages 215–220. Springer-Verlag, Berlin, 2003.

[19] J.A. Goldsmith. Unsupervised Learning of the Morphology of a Natural Language. *Computational Linguistics*, 27(2):153–198, 2001.

[20] N. Grabar and P. Zweigenbaum. Acquisition automatique de connaissances morphologiques sur le vocabulaire médical. In *TALN'99*, pages 175–184. Pascal Amsili, Ed., 1999.

[21] C. Hammarström. Unsupervised Learning of Morphology: Survey, Model, Algorithm and Experiments. Master's thesis, Department of Computer Science and Engineering, Chalmers University, 2007.

[22] H. Hammarström. A Naive Theory of Morphology and an Algorithm for Extraction. In R. Wicentowski and G. Kondrak, editors, *SIGPHON 2006: ACL Special Interest Group on Computational Phonology*, pages 79–88, 2006.

[23] N. Hathout. Acquisition morphologique à partir d'un dictionnaire informatisé. In *16ème conférence sur le Traitement Automatique des Langues Naturelles, TALN'09*, page 10p, 2009.

[24] N. Hathout. Acquisition of morphological families and derivational series from a machine readable dictionary. *CoRR*, abs/0905.1609, 2009.

[25] S. Helmut. Probabilistic part-of-speech tagging using decision trees. In *International Conference on New Methods in Language Processing*, September 1994.

[26] J. Hertz, A. Krogh, and R. Palmer. *Introduction to the Theory of Neural Computation.* Perseus Books, 1991.

[27] V. Hollink, J. Kamps, C. Monz, and M. De Rijke. Monolingual Document Retrieval for European Languages. *Information Retrieval*, 7(1-2):33–52, January 2004.

[28] D. Hull and G. Grefenstette. Stemming algorithms: A case study for detailed evaluation. *Journal of the American Society for Information Science*, 47(1):70–84, 1996.

[29] C. Jacquemin and E. Tzoukermann. NLP for term variant extraction: synergy between morphology, lexicon and syntax. In Tomek Strzalkowski, editor, *Natural Language Information Retrieval*, volume 7 of *Text, Speech and Language Technology*, pages 25–74. Kluwer Academic Publishers, Dordrecht/Boston/London, 1999.

[30] K. Kettunen, E. Airio, and K. Järveli. Restricted inflectional form generation in management of morphological keyword variation. *Information Retrieval*, 10(4–5):415–444, 2007.

[31] T. Korenius, J. Laurikkala, K. Jarvelin, and M. Juhola. Stemming and lemmatization in the clustering of finnish text documents. In *CIKM'04: Thirteenth ACM Conference on Information and Knowledge Management*, pages 625–633. ACM Press, 2004.

[32] Y. Lepage. Solving analogies on words: an algorithm. In *COLING-ACL'98*, pages 728–735, 1998.

[33] J.B. Lovins. Development of a Stemming Algorithm. *Mechanical translation and computational linguistics*, 11(1,2):23–31, 1968.

[34] C.D. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. The MIT Press, 1999.

[35] F. Moreau, V. Claveau, and P. Sébillot. Automatic Morphological Query Expansion Using Analogy-Based Machine Learning. In Lecture Notes in Computer Science, editor, *Advances in Information Retrieval*, volume 4425/2007 of *Computer Science*, pages 222–233. Springer-Verlag, Berlin / Heidelberg, 2007.

[36] C.D. Paice. Another stemmer. *SIGIR Forum*, 24(3):56–61, 1990.

[37] C.D. Paice. Method for Evaluation of Stemming Algorithms Based on Error Counting. *Journal of the American Society for Information Science*, 47(8):632–649, 1996.

[38] M.F. Porter. An algorithm for suffix stripping. *Program*, 40(3):211–218, 2006.

[39] J. Ross Quinlan. *C4.5: Programs for Machine Learning (Morgan Kaufmann Series in Machine Learning)*. Morgan Kaufmann, 1 edition, 1993.

[40] G. Souvay and J-M. Pierrel. LGeRM Lemmatisation des mots en Moyen Français. *Traitement Automatique des Langues*, 50(2):149–172, 2009.

[41] S. Tomlinson. Lexical and Algorithmic Stemming Compared for 9 European Languages with Hummingbird SearchServer. In *CLEF'03*, volume 3237, pages 286–300. Lecture Notes in Computer Science, Springer-Verlag, Berlin, 2003.

[42] A. Medina Urrea. Automatic Discovery of Affixes by means of a Corpus: A Catalog of Spanish Affixes. *Journal of Quantitative Linguistics*, 7(2):97–114, 2000.

[43] V. Vapnik. Principles of Risk Minimization for Learning Theory. In John E. Moody, Stephen Jose Hanson, and Richard Lippmann, editors, *Advances in Neural Information Processing Systems 4, [NIPS Conference, Denver, Colorado, USA, December 2-5, 1991]*, pages 831–838. Morgan Kaufmann, 1992.

[44] V. Vapnik. *The Statistical Learning Theory*. Springer, 1998.

[45] J. Vilares, M. A. Alonso, and M. Vilares. Extraction of complex index terms in non-English IR: A shallow parsing based approach. *Information Processing and Management*, 44(4):1517–1537, 2008.

[46] J. Vilares, D. Cabrero, and M. A. Alonso. Applying productive derivational morphology to term indexing of Spanish texts. In Alexander Gelbukh, editor, *Computational Linguistics and Intelligent Text Processing (CICLing'01)*, volume 2004 of *Lecture Notes in Computer Science*, pages 336–348. Springer-Verlag, Berlin-Heidelberg-New York, 2001.