Revista Vectores de Investigación
Journal of Comparative Studies Latin America
ISSN 1870-0128
ISSN online 2255-3371

Grigori Sidorov

N-GRAMAS SINTÁCTICOS Y SU USO EN LA LINGÜÍSTICA COMPUTACIONAL

APPLICATION OF SYNTACTIC N-GRAMS IN COMPUTATIONAL LINGUISTICS

Vol. 6 No. 6, 13-27 pp.

Grigori Sidorov

Centro de Investigaciónn en Computación (CIC), Instituto Politécnico Nacional, México

SNI III

Palabras claves: lingüística computacional, computación, n-gramas

N-gramas sintácticos y su uso en la lingüística computacional

APPLICATION OF SYNTACTIC N-GRAMS IN COMPUTATIONAL LINGUISTICS

ENVIADO 7-1-2013 **REVISADO** 24-1-2013
ACEPTADO 31-1-2013

RESUMEN En este artículo, estamos introduciendo un nuevo concepto que se utilizará en la lingüística computacional, se llama los n-gramas sintácticos: son n-gramas que se construyen siguiendo el árbol sintáctico. Es equivalente a introducir la información sintáctica en los métodos de aprendizaje automático, que siempre era un problema muy difícil. Discutimos los elementos que pueden formar estos n-gramas: palabras, clases gramaticales (*POS tags*), nombres de relaciones sintácticas, caracteres. Consideramos dos ejemplos de cómo se puede obtener los n-gramas sintácticos basándonos en un árbol sintáctico, tanto para el español como para el inglés. Adicionalmente, presentamos un modelo más utilizado de solución de problemas de la lingüística computacional, específicamente, el modelo de espacio vectorial. Al final, mostramos una aplicación de los n-gramas sintácticos para la tarea de atribución de autoría, en cuyo caso los resultados son mejores que los resultados de los métodos de línea base.

ABSTRACT This papers aims to introduce a new concept to be used in computational linguistics: the syntactic n-grams, which are formed following the paths in syntactic trees. The preceding is similar to entering the syntactic information into the automatic learning methods. Here, we discuss the elements of which the sn-grams can be formed, namely, words, grammatical features (POS tags), names of syntactic structures (SR tags) and characters. We consider two examples of how the syntactic n-grams can be obtained from a syntactic tree, both for English and Spanish. We also introduce a widely used model for problem representation in computational linguistics: the vector space model. Finally, we present an applications of the syntactic n-

grams for authorship attribution purposes, which obtains better results than the baseline methods.

1 Introducción

La lingüística computacional es un área importante dentro del campo de inteligencia artificial. De manera muy general, el propósito de la inteligencia artificial es modelar la inteligencia humana. Es decir, ¿qué es el comportamiento inteligente? ¿Cómo los humanos resuelven diariamente miles de problemas prácticos, en la gran mayoría de los casos sin equivocarse? Ésos son las preguntas importantes para la inteligencia artificial. La lingüística computacional analiza el papel del lenguaje en el comportamiento humano y estudia cómo construir los modelos del lenguaje de tal manera que sean entendibles para las computadoras.

Las computadoras son las herramientas modernas más importantes jamás creadas por la humanidad. Sin embargo la naturaleza de las computadoras consiste en una lógica binaria muy sencilla, ceros y unos, más las operación lógicas sobre ellos. ¿Cómo traducir un fenómeno tan complejo como un lenguaje humano a esa lógica tan simple? Para eso la lingüística computacional aprovecha tanto los conocimientos que tenemos sobre el lenguaje humano, como las herramientas que existen en el área de las ciencias de la computación y en las matemáticas: varios tipos de modelos formales, lenguajes de programación, etc.

De hecho es curioso, que muchos estudios modernos en el campo de la lingüística computacional, parecen cada vez más a otras áreas de la ciencia de la computación, tales como aprendizaje automático (clasificación automática o agrupamiento automático). En este caso la parte relacionada con la lingüística es como elegir las características que se presentan a los algoritmos de clasificación y agrupamiento. Entonces ¿por qué se dio este paso hacia el uso de los métodos formales? Porque con los avances de internet en la actualidad existe un gran número de textos libremente disponibles. Esos textos constituyen una excelente fuente para el aprendizaje de los sistemas automáticos. Pareciera ser que los sistemas automáticos modernos pueden hacer magia: toman los datos, y realizan las tareas de manera muy similar a un ser humano. En realidad, se basan en los métodos de aprendizaje automático.

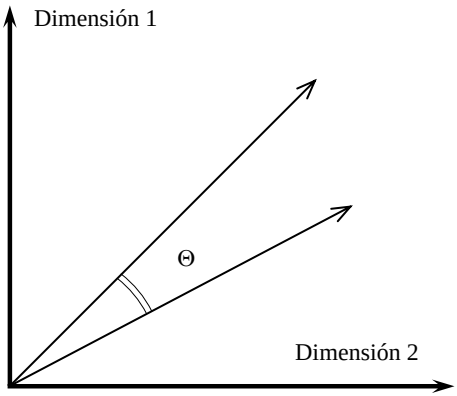
En el resto de este artículo se explica brevemente el modelo de espacio vectorial, que es uno de los modelos más utilizados en la lingüística computacional (Subtema 2). Se discute sobre los posibles valores que se pueden ser representados en este espacio, es decir, sobre n-gramas (Subtema 3), e introducimos un nuevo concepto de n-gramas sintácticos, que permiten utilizar la información sintáctica dentro de los métodos de procesamiento automático, tales como clasificación o agrupamiento (Subtema 4). En la Sección 5 presentamos una discusión sobre los posibles tipos de los n-gramas sintácticos. Sección 6 muestra un aplicación de los n-gramas sintácticos, se usa la tarea de atribución de autoría. Finalmente se presentan las conclusiones.

2 Modelo de espacio vectorial

El modelo de espacio vectorial es un modelo utilizado muy ampliamente en las ciencias de la computación. Su amplio uso se debe a la simplicidad del modelo y a su muy clara base conceptual que corresponde a la intuición humana en el procesamiento de información. Realmente la idea detrás del modelo es muy sencilla y es una respuesta a la pregunta de cómo podemos comparar los objetos de manera formal. Parece que la única manera de representar los objetos es utilizar la representación con los rasgos (características) y sus valores. Es una idea universal, y hasta parece que es la única manera posible de trabajar con los objetos de manera formal. Quizá algo similar pero distinto en algunos aspectos sería el uso de las memorias asociativas.

Bueno, ahora suponemos que ya representamos dos objetos (o más) eligiendo las características que ellos tienen y los valores de esas características. Para tenerlo completamente claro, consideraremos un ejemplo. Supongamos que queremos comparar dos libros. ¿Qué características podemos elegir? Eso en gran medida depende de nuestras necesidades, es decir, no existe un conjunto único de las características de los objetos; sin embargo, hay algunas características más comunes que otras, por ejemplo, en el caso de los libros, el número de páginas sería una característica importante para muchos propósitos. También podrían ser las características como color de la portada, autor, editorial, el perfil sociológico de las personas a quienes gustó este libro, etc. En el caso de número de páginas, el valor de esta característica sería numérico, es decir, un número. En caso de la editorial, sería una lista posible de las editoriales. En caso de los perfiles sociológicos, los valores de esa característica serían un poco más difíciles de representar, por ejemplo, “albañiles de 30 a 40 años”, o “profesores universitarios de 40 a 50 años”, etc. Nótese que la selección tanto de las características como de sus valores queda a elección propia, es decir, cada quien construye su modelo; y de su aplicación práctica depende el si es útil o no.

Figura 1. Ejemplo del espacio vectorial: similitud entre vectores



Fuente: Elaboración propia

Ahora bien, se tiene la representación de dos objetos en términos de las características y de sus valores. ¿Cuál es el paso siguiente? ¿Qué se necesita para construir un modelo de espacio vectorial? ¿Debe ser algo muy complejo? Realmente, no es así, de hecho ya se tiene este modelo. Es un espacio de N dimensiones, y cada dimensión en este espacio corresponde a una de las características: tenemos tantas dimensiones cuantas características tiene el objeto en nuestro modelo. Se puede imaginar una dimensión como un eje, en el cual podemos marcar los valores de esa característica/dimensión. Si los valores son numéricos la interpretación es muy clara. Si los valores son naturalmente ordenados, como, por ejemplo, los intervalos de la edad, también está claro cómo tratarlos. Si los valores no son relacionados, como por ejemplo el color de la portada de un libro, la solución más fácil es construir un orden aleatorio.

El siguiente paso está relacionado con la pregunta: ¿y dónde están los vectores, por qué es espacio vectorial? Como ya se mencionó, cada objeto es un conjunto de los valores de sus características, lo que corresponde a exactamente un punto en un espacio de N dimensiones. Este punto exactamente corresponde a un vector n-dimensional, que empieza en el punto (0,0,0,..) en este mismo espacio. Por ejemplo, un libro de 100 páginas y con color de la portada *rojo*, comparado con otro libro de 50 páginas y con el color de la portada *verde*. Este es espacio de dos dimensiones: número de páginas en color de la portada, y cada libro es un punto con las coordenadas (100, *rojo*) y (50, *verde*). Nótese que tenemos que elegir, cuál de las dos, *rojo* o *verde*, corresponde a la coordenada cero; pero esa elección no afectará las consideraciones futuras.

Otra pregunta es cómo podemos representar formalmente los espacios vectoriales. Como ya se dijo, cada objeto es un conjunto de características y algunos de sus valores, por ejemplo, uno de los libros se presenta como X=[*(número de páginas=100), (color de portada=rojo)*], y el otro Y=[*(número de páginas=50), (color de portada=verde)*]. Surgen dos preguntas: 1) ¿si hay que repetir cada vez el nombre de la característica? y 2) ¿qué se puede hacer en caso que algún objeto no tenga alguna característica?

La respuesta de la primera pregunta es muy sencilla, simplemente podemos meter toda la información en una tabla, donde las columnas van a corresponder a los objetos, y las filas a las características (o puede ser al revés, eso no cambia el modelo), véase Tabla 1. En este sentido, representación tabular (como una matriz) y representación vectorial es lo mismo.

Tabla 1. Ejemplo de representación tabular del espacio vectorial

	Libro X	**Libro Y**
Dimensión 1: número de páginas	100	50
Dimensión 2: color de portada	rojo	verde

Fuente: Elaboración propia

En este sentido, sólo con saber a qué columna pertenece un valor, se sabe a qué característica corresponde este valor; es decir, se sabe simplemente por su posición.

Igualmente sencilla es la respuesta para la segunda pregunta, qué hacer en caso de que el objeto simplemente no tenge una característica dada: se ponen los valores iguales a cero en el lugar correspondiente. En todos los cálculos posteriores esos valores no afectarán el resultado, siendo iguales a cero.

Nótese que la representación tabular es conceptualmente lo mismo que el modelo de espacio vectorial *per se*: las columnas corresponden a las dimensiones. La única diferencia es que en este caso no es tan natural el uso de los conceptos geométricos.

Ahora ¿qué ventajas da el concepto de espacio vectorial? Ya lo tenemos, ¿y qué? Resulta que se puede utilizar la metáfora de espacio para calcular la similitud entre objetos, es decir, comparar los objetos, basándonos en puras ideas geométricas muy sencillas, no más allá del teorema de Pitágoras.

Entonces, cada objeto es un vector en un espacio de N dimensiones. ¿Cómo comparar esos vectores? La idea geométrica es que los vectores que tienen más o menos la misma dirección se parecen entre sí. Hablando de manera más formal, menor es el ángulo entre esos vectores, mayor es la similitud. Es muy claro e intuitivo en un espacio de dos dimensiones. En un espacio con mayor número de dimensiones es más difícil imaginar esa similitud, por eso se sugiere al lector siempre considerar los ejemplos en un espacio de dos dimensiones; tomando en cuenta que en un espacio con mayor número de dimensiones las ideas serían exactamente iguales. Y finalmente para expresar más formalmente esta similitud, se utiliza la medida del coseno del ángulo entre los vectores: menor el ángulo mayor del coseno, es decir, la similitud entre vectores (y objetos).

La única consideración es que antes (o posiblemente después) del cálculo de coseno, los vectores se normalizan aplicando lo que se conoce como la norma euclidiana. La norma euclidiana convierte cada vector a un vector de longitud uno. La norma euclidiana consiste en la división del vector entre su longitud, denominado $\|v\|$.

$$\|v\| = \sqrt{\sum_{k=0}^{n-1} v_k^2}$$

En caso de dos dimensiones, es claramente el teorema de Pitágoras: $||v||=sqrt(v_0^2+v_1^2)$, donde v_0 y v_1 son valores del vector en el eje 0 y eje 1.

Para calcular la similitud de coseno entre dos vectores, *va* y *vb*, al final se tendrá que dividir el resultado entre la multiplicación de sus longitudes, $||va||*||vb||$.

Y finalmente, la fórmula para el cálculo del coseno es un producto punto de ambos vectores, es decir, se suman las multiplicaciones de los valores de vectores en cada dimensión. En dos dimensiones 0 y 1 y para los vectores *va* y *vb*, eso sería: dot_product= $va_0*vb_0 + va_1*vb_1$. No hay que olvidar aplicar al final la norma euclidiana. De manera general:

$$\cos\theta = \sum_{k=0}^{n-1} va_k * vb_k \Big/ \|va\| * \|vb\|$$

En este caso, la medida de coseno indica que tan parecido son los vectores *va* y *vb.* En caso de los valores positivos, coseno está en el rango de 0 a 1.

Las ideas que se presentan son muy sencillas; sin embargo, permiten comparar cualquier tipo de objetos, utilizando el modelo de espacio vectorial, la correspondiente metáfora espacial y las ideas geométricas muy sencillas.

3 Modelo de espacio vectorial para comparación de documentos

Ahora bien, ya sabemos qué es un modelo de espacio vectorial. Vamos a ver cómo se aplica para la comparación de documentos/textos. Es decir, nuestros objetos que queremos comparar son documentos. Entonces, el problema es cómo elegir las características que tienen documentos. Igual que siempre con el modelo de espacio vectorial tenemos muchas opciones, y depende de nosotros que características vamos a considerar importantes.

La idea más sencilla es utilizar las palabras como características de documentos. Normalmente se aplican algunos procedimientos adicionales, como por ejemplo, la lematización, es decir, todas las formas se sustituyen por sus lemas (formas normalizadas, por ejemplo, *trabajaron, trabajamos, trabajen* etc. tienen el lema *trabajar*). También muy a menudo del cálculo de la similitud se excluyen las palabras auxiliares (también llamados *stop words*), como preposiciones o artículos, porque su presencia en un documento no dice nada del documento como tal, su presencia es determinada por las características del lenguaje. Es usual, con excepción si tengamos una tarea específica que requiera presencia de esas palabras, por ejemplo, análisis de estilo.

Si las características son las palabras, ¿qué podemos utilizar como sus valores? Intuitivamente, puede ser algo relacionado con frecuencia de la palabra. En este sentido, más frecuente es la palabra, más importante es esta palabra para el documento. No todo es tan sencillo, pero la idea es ésa.

La frecuencia de la palabra se denomina TF (*term frequency,* frecuencia: cuantas veces una palabra ocurre en un documento) y normalmente se combina con otra medida, que se llama IDF (*inverse document frequency,* frecuencia inversa en documentos). La intuición detrás del IDF se relaciona con lo siguiente: si una palabra se encuentra en todos los documentos de nuestra colección, entonces esta palabra es incapaz de distinguir entre los documentos, y por lo tanto no nos sirve. Y al contrario, si una palabra se encuentra exactamente en un documento en nuestra colección, es una palabra muy útil para cálculos de similitud (o recuperación de información, que es un caso particular de cálculo de similitud). El IDF se calcula para cada palabra en una colección dada, es decir, depende de la colección, y pondera el TF de una palabra dada en cada documento: normalmente se multiplican. La medida TF-IDF muy a menudo se utiliza como el valor de las palabras para comparar los documentos.

Nótese que en caso de utilizar las palabras como características se pierde la

información sobre las relaciones sintácticas entre palabras. Las palabras se convierten en lo que se llama bolsa de palabras (*bag of words*). Para muchas tareas esa pérdida de información es aceptable. Más adelante en el artículo se propone una posible solución para evitar esa pérdida de información.

¿Qué más aparte de las palabras podrían ser las características de documentos? Quizá, es difícil inventarlo rápido desde cero, pero el concepto como tal es muy sencillo. Se trata de n-gramas. La sencillez de los modelos que describimos es algo que ya hemos afirmado sobre el modelo de espacio editorial, y creemos que podemos convencer al lector de que realmente es así.

N-gramas tradicionales son secuencias de elementos tal como aparecen en un documento, por ejemplo, (Jurafsky y Martin, 2009; Sidorov, Velásquez, *et al.*, 2012; Khalilov y Fonollosa, 2009; Habash, 2004; Agarwal, Biads, Mckeown, 2009; Cheng, Greaves y Warren, 2006; Baayen, Tweedie y Halteren, 1996). En este caso N indica cuantos elementos debemos tomar, es decir, la longitud de la secuencia o de n-grama por ejemplo, existen bigramas, trigramas, cuatrigramas (2-gramas, 3-gramas, 4-gramas), etc. De esa manera, si se va a hablar de unigramas, es decir, de n-gramas construidos de un solo elemento, es lo mismo que hablar de palabras.

Otro grado de libertad es el tipo de elementos que están formando n-gramas. Pueden ser lemas o palabras, pero también pueden ser clases gramaticales (en inglés, POS, *part of speech*), como sustantivos, verbos, etc., posiblemente con sus características gramaticales más detalladas, por ejemplo, una etiqueta como VIP1S, podría significar "verbo, indicativo, presente, primera persona, singular". Podemos tener n-gramas de ese tipo de etiquetas. En los últimos años se utiliza mucho los n-gramas de caracteres, es decir secuencias de caracteres tomadas de un texto. Curiosamente, para algunas tareas dan buenos resultados. Su interpretación lingüística no está clara, y es objeto de estudios.

Vamos a ver un ejemplo de los n-gramas tradicionales de palabras. De la frase: Juan lee un libro interesante podemos sacar los siguientes 2-gramas: Juan lee, lee un, un libro, libro interesante. O los siguientes 3-gramas: Juan lee un, lee un libro, un libro interesante, etc. Podemos sustituir cada palabra por su lema o por su clase gramatical, y tener los n-gramas correspondientes. Como se puede observar, el procedimiento es muy sencillo, pero se usa con gran éxito en los sistemas de lingüística computacional.

Y si utilizamos los n-gramas como características, ¿qué valores pueden tener? Igual que en el caso de las palabras, son los valores relacionados con TF-IDF. Nótese que las frecuencias de n-gramas usualmente son mucho menores que frecuencias de las palabras, es decir los n-gramas aparecen mucho menos en los textos. Es lógico, porque realmente estamos requiriendo la aparición al mismo tiempo de dos o más palabras seguidas, lo que es un evento mucho menos probable.

4 Concepto de n-gramas sintácticos

Como ya se mencionó, la idea principal de las características formales aplica-

bles en la lingüística computacional se relaciona con el uso de los n-gramas. Los N-gramas tradicionales descritos arriba ignoran la información sintáctica, son "bolsas de palabras". Entonces, la pregunta es, ¿cómo se puede seguir utilizando la técnica de n-gramas, de la cual se sabe que da muy buenos resultados, y al mismo tiempo utilizar la información sintáctica? La respuesta que se propone en este artículo es la manera especial de obtención de n-gramas. Se sugiere sacar los n-gramas siguiendo las rutas en los árboles sintácticos. Intuitivamente, se sigue teniendo en n-gramas, pero se evita el ruido introducido por la estructura superficial del lenguaje. Este tipo de ruido puede aparecer, porque en el nivel superficial las palabras no relacionadas pueden aparecer juntas. Este fenómeno se combate si se siguen las relaciones sintácticas reales.

Considérense dos ejemplos, ambos están tomados de un libro de Julio Verne. El primer ejemplo está en inglés, y el segundo está en español. Para ambos ejemplos se construyó el árbol sintáctico, es decir, la información sintáctica.

Para el primer ejemplo, en inglés, se usó el Stanford *parser*[1] (Marneffe, MacCartney, Manning, 2006). Primero se presenta la salida del *parser,* tal como lo genera el propio programa, y después en la figura 2, se presenta el árbol de dependencias utilizando las flechas de varios niveles, es decir, empezando de la raíz de la oración, y se baja mientras se sigue la ruta sintáctica.

The wildest cheering resounded on all sides; the name of Ferguson was in every mouth. (Lit.: "Un fuerte aplauso sonó por todos lados; el nombre de Ferguson estuvo en cada boca.")

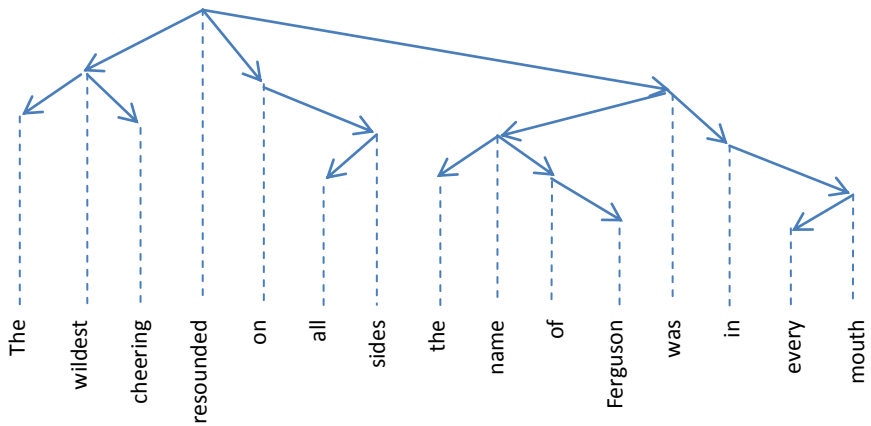
En la salida directa del *parser*, se presenta la información a nivel de constituyentes, donde mayor sangría en la línea corresponde a mayor profundidad de cada palabra. Sin embargo, para nuestros propósitos, presenta más interés la otra parte de la salida, donde se pueden observar las parejas de palabras y la relación sintáctica entre ellas. De esa segunda parte de la salida se puede construir directamente el árbol sintáctico, tal como se presenta en las figuras 2a y 2b. En las figuras 2a y 2b se presentan los mismos árboles sintácticos, siendo la única diferencia, que en la figura 2b sobre cada flecha aparece el nombre de la relación sintáctica correspondiente.

```
(ROOT
 (S
  (S
   (NP
    (NP (DT The) (NN wildest))
    (VP (VBG cheering)))
   (VP (VBD resounded)
    (PP (IN on)
     (NP (DT all) (NNS sides)))))
  (: ;)
  (S
```

[1] Un *parser* es un programa que construyen los árboles sintácticos. Usualmente se basan en algunas gramáticas formales de varios tipos.

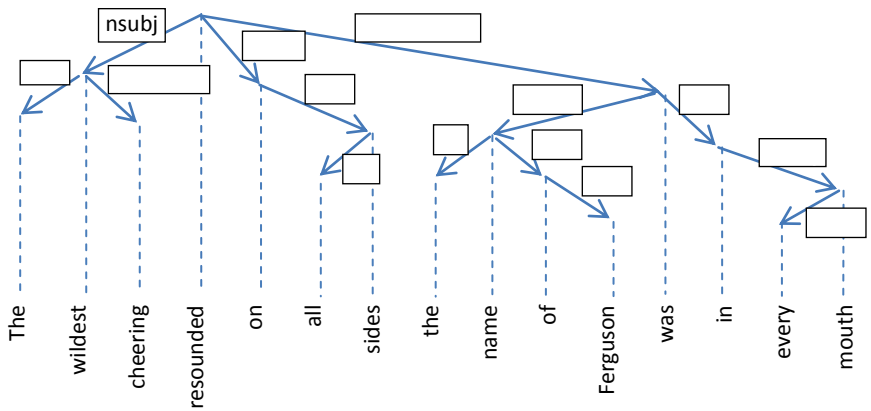
(S
(NP
(NP (DT the) (NN name))
(PP (IN of)
(NP (NNP Ferguson))))
(VP (VBD was)

Figura 2a. Ejemplo de un árbol sintáctico en inglés con etiquetas



Fuente: Elaboración propia.

Fig. 2b. Ejemplo de un árbol sintáctico en inglés con etiquetas

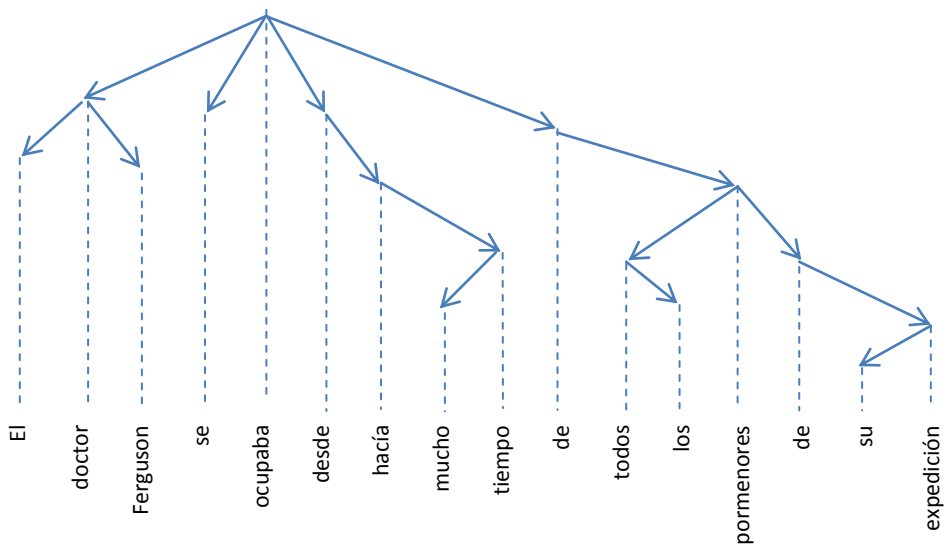


Fuente: Elaboración propia

(PP (IN in)
(NP (DT every) (NN mouth)))))
(. .)))
det(wildest-2, The-1)

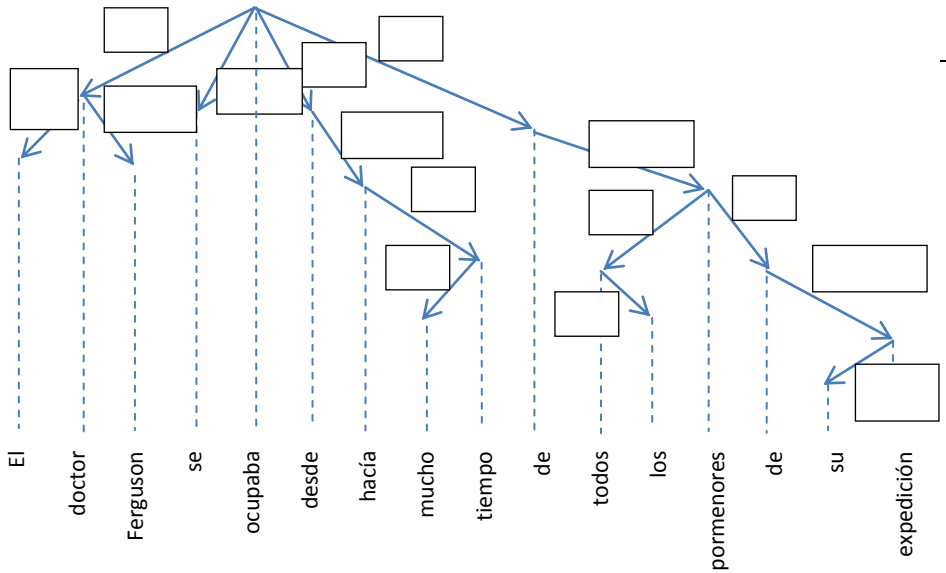
nsubj(resounded-4, wildest-2)
partmod(wildest-2, cheering-3)
root(ROOT-0, resounded-4)
prep(resounded-4, on-5)
det(sides-7, all-6)
pobj(on-5, sides-7)
det(name-10, the-9)
nsubj(was-13, name-10)
prep(name-10, of-11)
pobj(of-11, Ferguson-12)
parataxis(resounded-4, was-13)
prep(was-13, in-14)
det(mouth-16, every-15)
pobj(in-14, mouth-16)

Figura 3a. Ejemplo de un árbol sintáctico en español



Fuente: Elaboración propia

Figura 3b. Ejemplo de un árbol sintáctico en español con etiquetas



Fuente: Elaboración propia

Es bastante obvio después de la explicación anterior, qué n-gramas se puede obtener de esta oración. Por ejemplo, se pueden obtener los siguientes 2-gramas: *resounded-wildest, wildest-cheering, wildest-the, resounded-on, on-sides, resounded-was, was-name, name-of, of-Ferguson, was-in, in-mouth, mouth-every*. Y los siguientes 3-grams: *resounded-wildest- cheering, re-sounded-wildest-the, resounded-was-name, resounded-was-in*, etc.

Para analizar el ejemplo en español se utilizó el *parser* del sistema FreeLing (Padró, Collado, *et. al.,* 2010). Las relaciones sintácticas se muestran en el resultado de análisis con sangría, es decir, los bloques con la misma sangría tienen la misma palabra principal (si no apareció otra posible palabra principal entre esas palabras). Se puede observar que el *parser* se equivocó y colocó el grupo "de su expedición" como dependiente del verbo "hacer" en lugar del verbo "ocuparse". Los *parsers* pueden cometer este tipo de errores con algunos tipos de la ambigüedad sintáctica difícil de resolver de manera automática. Sin embargo, aun así, en muchos casos eso no afectará significativamente el funcionamiento del sistema, dado que la gran mayoría de relaciones se establecieron correctamente. En el árbol del figura 3, se corrigió este error del *parser*.

El doctor Ferguson se ocupaba desde hacía mucho tiempo de todos los pormenores de su expedición.

```
grup-verb/top/(ocupaba ocupar VMII1S0 -) [
  morfema-verbal/es/(se se P0000000 -)
```

```
 sn/subj/(doctor doctor NCMS000 -) [
  espec-ms/espec/(El el DA0MS0 -)
  w-ms/sn-mod/(Ferguson ferguson NP00000 -)
 ]
 prep/modnomatch/(desde desde SPS00 -)
 grup-verb/modnomatch/(hacía hacer VMII1S0 -) [
  sn/cc/(tiempo tiempo NCMS000 -) [
   espec-ms/espec/(mucho mucho DI0MS0 -)
   sp-de/sp-mod/(de de SPS00 -) [
    sn/obj-prep/(pormenores pormenor NCMP000 -) [
     espec-mp/espec/(todos todo DI0MP0 -) [
      j-mp/espec/(los el DA0MP0 -)
     ]
    ]
   ]
   sp-de/sp-mod/(de de SPS00 -) [
    sn/obj-prep/(expedición expedición NCFS000 -) [
     espec-fs/espec/(su su DP3CS0 -)
    ]
   ]
  ]
  F-term/term/(. . Fp -)
 ]
]
```

La metodología de obtención de los n-gramas sintácticos es similar en el caso inglés y español.

5 Tipos de n-gramas sintácticos

Ahora bien, ya se sabe cómo obtenerlos, ahora se analizan qué tipos de n-gramas sintácticos existen. Es decir, qué tipo de elementos pueden formar los n-gramas en este caso. Obviamente pueden ser palabras, como en los ejemplos mencionados anteriormente.

De la misma manera en lugar de palabras se puede utilizar la información gramatical correspondiente, por ejemplo, en el caso en español (figura 3), NCFS000 o VMII1S0, etc. El término correspondiente en inglés es *POS tags*, o *POS.*

También hay una nueva posibilidad de utilizar como los elementos de n-gramas los nombres de las relaciones sintácticas que están presentes en el árbol sintáctico, por ejemplo, *nsubj* o *pobj*, en casos en inglés (figura 2). Los

nombres de relaciones sintácticas están presentes en las figuras 2b y 3b, sobre las flechas.

La última posibilidad que está presente para los n-gramas tradicionales es utilizar los caracteres. También se pueden utilizar para n-gramas sintácticos: se pueden sacar de la misma manera que para los n-gramas tradicionales, pero utilizando los bigramas o trigramas sintácticos. Es cuestión de futuras investigaciones determinar si este tipo de n-gramas es útil.

Resumiendo, se puede decir que existen n-gramas sintácticos de:

- Palabras.
- POS tags.
- Relaciones sintácticas.
- Caracteres.

6 Ejemplo de aplicación: atribución de autoría

Se han desarrollado algunos experimentos (Sidorov, Velasquez, *et al.,* 2012), para probar la utilidad del concepto de n-gramas sintácticos. Esencialmente, se consideró la tarea de atribución de autoría, es decir, hay textos de los cuales se conocen sus autores, y en otros se debe determinar su autor. En este caso se consider´ó un corpus de textos para tres autores.

La tarea de atribución de autoría es claramente una tarea de clasificación. Se utilizan como características los n-gramas tradicionales y los n-gramas sintácticos de tamaños de 2 a 5, y una herramienta que permite fácilmente aplicar varios algoritmos de clasificación: WEKA (Hall, Frank, *et al.*, 2009).

Se utiliza un corpus compuesto de obras de los autores: *Booth Tarkington, George Vaizey, Louis Tracy* (escribían en inglés en el siglo XIX). El corpus fue compuesto por cinco novelas de cada autor para el entrenamiento, 11 MB, y de tres obras de cada autor para la clasificación, 6 MB.

Se usan perfiles de varios tamaños. El término perfil quiere decir que se utiliza el número correspondiente de los n-gramas más frecuentes, por ejemplo, para el perfil de 400, nada más 400 n-gramas que tienen mayor frecuencia en el corpus de entrenamiento, etc.

Se utiliza el método de clasificación estándar que se llama máquinas de vectores de soporte (*support vector machines*, en inglés).

Los resultados de clasificación (atribución de autoría) se presentan en la tabla 1 para los 2-gramas, y en la tabla 2 para los 3-gramas.

Tabla 1. Resultados de atribución de autoría para 2-gramas

Tamaño de perfil	Características			
	n-gramas sintácticos	n-gramas de POS tags	n-gramas de caracteres	n-gramas de palabras
400	100%	90%	90%	86%
1,000	100%	95%	95%	86%
4,000	100%	ND	90%	86%

7,000	100%	ND	ND	86%
11,000	100%	ND	ND	89%

Fuente: Elaboración propia

Tabla 2. Resultados de atribución de autoría para 3-gramas

Tamaño de perfil	Características n-gramas sintácticos	n-gramas de POS tags	n-gramas de caracteres	n-gramas de palabras
400	100%	90%	76%	81%
1,000	100%	90%	86%	71%
4,000	100%	100%	95%	95%
7,000	100%	100%	90%	90%
11,000	100%	95%	100%	90%

La leyenda ND quiere decir que no se encontraron tantos n-gramas para el perfil específico.
Fuente: Elaboración propia

Como se puede observar, el método basado en los n-gramas sintácticos obtuvo los mejores resultados. Sin embargo, cabe mencionar que el problema considerado tiene un *top-line* (facilidad de obtener los resultados) bastante alto, dado que se tienen muchos datos de entrenamiento y solamente se utilizan tres clases (autores).

7 Conclusiones

En este artículo, se introduce un nuevo concepto que se utilizará en la lingüística computacional; se llama los n-gramas sintácticos. Su uso equivale a introducir la información sintáctica en los métodos de aprendizaje automático, que siempre era un problema muy difícil. Se discutieron los elementos que pueden formar estos n-gramas: palabras, clases gramaticales (*POS tags*), nombres de relaciones sintácticas, caracteres.

Se consideraron dos ejemplos de cómo se puede obtener los n-gramas sintácticos basándonos en un árbol sintáctico, tanto para el español como para el inglés.

Adicionalmente, presentamos un modelo más utilizado para solución de problemas de la lingüística computacional, específicamente, el modelo de espacio vectorial.

Se mostró una aplicación de los n-gramas sintácticos para la tarea de atribución de autoría, en cuyo caso los resultados son mejores que los resultados de los métodos de línea base.

BIBLIOGRAFÍA

JURAFSKY, Daniel, MARTIN, James (2009) *Speech and Language Processing,* Prentice Hall.

GELBUKH, Alexander, SIDOROV, Grigori (2010) *Procesamiento automático del español con enfoque en recursos léxicos grandes*, México, Instituto Politécnico Nacional.

SIDOROV, Grigori, VELASQUEZ, Francisco, STAMATATOS, Efstathios, GELBUKH, Alexander, CHANONA-HERNÁNDEZ, Liliana (2012) "Syntactic Depen-

dency-based N-grams as Classification Features", *LNAI*, vol. 7630, 1–11.

KHALILOV, Maxium, FONOLLOSA, José (2009) "N-gram-based Statistical Machine Translation versus Syntax Augmented Machine Translation: comparison and system combination", *Proceedings of the 12th Conference of the European Chapter of the ACL*, 424–432.

HABASH, Nizar (2004) "The Use of a Structural N-gram Language Model in Generation-Heavy Hybrid Machine Translation", *Natural Languaje generation*, Anja Belz, Roger Evans, Paul Piwek (edición), Saarbrücken, Springer, 61–69.

AGARWAL, Apoorry, BIADS, Fadi, MCKEOWN, Kathleen (2009) "Contextual Phrase-Level Polarity Analysis using Lexical Affect Scoring and Syntactic N-grams", *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL)*, 24–32.

CHENG, Winnie, GREAVES, Chris, WARREN, Martin (2006) "From n-gram to skipgram to concgram", *International Journal of Corpus Linguistics,* vol. 11, N° 4, 411-433.

BAAYEN, Harald van, TWEEDIE, Fiona, HALTEREN, Hans (1996) "Outside The Cave of Shadows: Using Syntactic Annotation to Enhance Authorship Attribution", *Literary and Linguistic Computing*, vol. 11, N° 3, 121–131.

STAMATATOS, Efstathios (2009) "A survey of modern authorship attribution methods", *Journal of the American Society for information Science and Technology,* vol. 60, N° 3, 538–556

HALL, Mark, FRANK, Eibe, HOLMES, Geoffrey, PFAHRINGER, Bernhard, REUTEMANN, Peter, WITTEN, Ian (2009) "The WEKA Data Mining Software: An Update", *SIGKDD Explorations*, vol. 11, N° 1, 10-18.

MARNEFFE, Marie-Catherine de, MACCARTNEY, Bill, MANNING, Christopher (2006) "Generating Typed Dependency Parses from Phrase Structure Parses", *Proceding of LREC,* 449-454.

PADRÓ, Lluís, COLLADO, Miquel, REESE, Samuel, LLOBERES, Marina, CASTELLÓN, Irene, (2010) "FreeLing 2.1 Five Years of Open-Source Language Processing Tools", *Proceedings of 7th Language Resources and Evaluation Conference (LREC 2010)*, ELRA La Valletta, Malta.