Polibits

ISSN: 1870-9044

polibits@nlp.cic.ipn.mx

Instituto Politécnico Nacional

México

Sidorov, Grigori

N-gramas sintácticos no-continuos

Polibits, vol. 48, 2013, pp. 69-78

Instituto Politécnico Nacional

Distrito Federal, México

N-gramas sintácticos no-continuos

Grigori Sidorov

***Resumen*—En este artículo presentamos el concepto de los n-gramas sintácticos no-continuos. En nuestros trabajos previos hemos introducido un concepto general de los n-gramas sintácticos, es decir, los n-gramas que se construyen siguiendo las rutas en un árbol sintáctico. Su gran ventaja consiste en que permiten introducir información puramente lingüística (sintáctica) en los métodos computacionales de aprendizaje automático. Su desventaja está relacionada con la necesidad de realizar el análisis sintáctico automático previo. También hemos demostrado que la aplicación de los n-gramas sintácticos en la tarea de atribución de autoría da mejores resultados que el uso de los n-gramas tradicionales. Sin embargo, en dichos trabajos sólo hemos considerado los n-gramas sintácticos continuos, es decir, durante su construcción no se permiten bifurcaciones en las rutas sintácticas. En este artículo, estamos proponiendo a quitar esta limitación, y de esa manera considerar todos los sub-árboles de longitud *n* de un árbol sintáctico como los n-gramas sintácticos no-continuos. Cabe mencionar que los n-gramas sintácticos continuos son un caso particular de los n-gramas sintácticos no-continuos. El trabajo futuro debe mostrar qué tipo de n-gramas es más útil y para qué tareas de PLN. Se propone la manera formal de escribir un n-grama sintáctico no-continuo usando paréntesis y comas, por ejemplo, "*a b [c [d, e], f]*". También presentamos en este artículo ejemplos de construcción de n-gramas sintácticos no-continuos para los árboles sintácticos obtenidos usando FreeLing y el *parser* de Stanford.**

***Palabras clave*—Modelo de espacio vectorial, n-gramas, n-gramas sintácticos continuos, n-gramas sintácticos no-continuos.**

Non-continuous Syntactic N-grams

***Abstract*—In this paper, we present the concept of non-continuous syntactic n-grams. In our previous works we introduced the general concept of syntactic n-grams, i.e., n-grams that are constructed by following paths in syntactic trees. Their great advantage is that they allow introducing of the merely linguistic (syntactic) information into machine learning methods. Certain disadvantage is that previous parsing is required. We also proved that their application in the authorship attribution task gives better results than using traditional n-grams. Still, in those works we considered only continuous syntactic n-grams, i.e., the paths in syntactic trees are not allowed to have bifurcations. In this paper, we propose to remove this limitation, so we consider all sub-trees of length *n* of a syntactic tree as non-continuous syntactic n-grams. Note that continuous syntactic n-grams are the particular case of non-continuous syntactic n-grams. Further research should show which n-grams are more useful and in which NLP tasks. We also propose a formal manner of writing down (representing) non-continuous syntactic n-grams using parenthesis and commas, for example, "*a b [c [d, e], f]*". In this paper, we also present examples of construction of non-continuous syntactic n-grams on the basis of the syntactic tree of the FreeLing and the Stanford parser.**

***Index Terms*—Vector space model, n-grams, continuous syntactic n-grams, non-continuous syntactic n-grams.**

Manuscrito recibido 12 de junio de 2013. Manuscrito aceptado para su publicación 23 de septiembre de 2013.

Grigori Sidorov trabaja en el Centro de Investigación en Computación (CIC), Instituto Politécnico Nacional (IPN), Av. Juan de Dios Bátiz, s/n, esq. Othón de Mendizábal, Zacatenco, 07738, México DF, México (e-mail: www.cic.ipn.mx/~sidorov).

I. Introducción

En el análisis automático de lenguaje natural (procesamiento de lenguaje natural, PLN) y en la lingüística computacional [1] cada vez son más populares los métodos relacionados con el aprendizaje automático computacional (*machine learning*, en inglés). Aplicando estos métodos se obtienen resultados cada vez mejores [2], [3].

La intención principal detrás de la aplicación de los métodos de aprendizaje automático es tratar de modelar la formulación de hipótesis por parte de los lingüistas y su posterior verificación. En este caso se sustituye la intuición humana por las grandes cantidades de datos textuales, posiblemente con marcas adicionales hechas manualmente, y por métodos sofisticados de aprendizaje, que se basan en las matemáticas y en las estadísticas. De ninguna manera se pretende sustituir a los lingüistas en el proceso de investigación, sino desarrollar herramientas que puedan ser útiles para ellos. Por otro lado, la aplicación de métodos de aprendizaje automático permite la evaluación exacta de las hipótesis, la reproducción de los resultados y hasta cierto punto convierte a la lingüística computacional en una ciencia más exacta. En este sentido, algunas partes de la lingüística computacional ya requieren procedimientos empíricos, cuando la hipótesis formulada se verifica utilizando los experimentos computacionales basados en datos, y no sólo en la intuición de hablantes nativos o del propio experimentador.

Los métodos más utilizados en la lingüística computacional son en su mayoría métodos supervisados, es decir, se utilizan datos marcados manualmente para realizar entrenamiento. Otra posibilidad está relacionada con los métodos no supervisados, cuando el propio sistema debe aprender directamente de los datos. Claro está que es mucho más difícil utilizar los métodos no supervisados, porque en este caso en ningún lado se encuentra una intervención humana; normalmente para poder aplicar estos métodos se requiere una gran cantidad de datos.

Por lo mencionado anteriormente, en la lingüística computacional ya son aplicables los conceptos de precisión y especificidad (*precision* y *recall*, en inglés), que miden la viabilidad de una hipótesis de manera formal. Igual que el concepto de línea base (*baseline*), que corresponde al método

comúnmente aceptado del estado del arte que resuelve el mismo problema, y que debe ser superado por la hipótesis propuesta. Normalmente la línea base es un método no muy sofisticado. Dado que estamos hablando de datos marcados por seres humanos de manera manual y que se usan para la verificación del método, se utiliza el concepto de estándar de oro (*gold standard*). Como es marcado de manera manual, se supone que no tiene errores (de aquí viene la idea que es de "oro"), y si el sistema lo puede alcanzar, entonces el sistema funciona realmente muy bien. Una cuestión interesante está relacionada con la concordancia entre los juicios humanos, es decir, si los propios seres humanos marcan de manera diferente algún fenómeno lingüístico, entonces no podemos esperar que la máquina resuelva correctamente este mismo fenómeno. Aquí viene el concepto de la línea tope (*top line*, en inglés). Por lo mismo, es recomendable usar varios evaluadores, y no sólo uno, para tener juicios mejor fundamentados y menos sesgados.

Para realizar los experimentos se utiliza comúnmente la técnica llamada validación cruzada de *k* volúmenes (k-*fold cross validation*), donde *k* tiene un valor numérico, normalmente igual a 10. La técnica consiste en dividir todos los datos en 10 (o *k*) volúmenes. Primero se utiliza el volumen 1 para la evaluación de los resultados del sistema, y los volúmenes 2 a 10 para el entrenamiento, después se elige el volumen 2 para la evaluación, y los otros nueve volúmenes para el entrenamiento, y así sucesivamente. De esta manera el sistema es evaluado 10 veces sobre datos diferentes y se entrena 10 veces sobre datos algo diferentes, finalmente se toma el promedio de las 10 evaluaciones. Nótese que es muy importante no realizar la evaluación sobre los mismos datos, sobre los cuales se hizo el entrenamiento: por eso es la división en *k* volúmenes.

Para realizar todos los procedimientos descritos, tenemos que representar el problema de manera formal. El modo más utilizado de representación es el modelo de espacio vectorial [2], [4]. Se representa el problema como un problema de clasificación automática en un espacio, que es precisamente el espacio vectorial. Los objetos se representan como conjuntos de valores de características, es decir, a cada objeto le corresponde un vector de dichos valores (de aquí el término "espacio vectorial"). Eso significa que cada objeto es un punto en el espacio multidimensional de características. Es muy fácil imaginar este modelo para el caso de dos características (dos dimensiones). Para un mayor número de dimensiones simplemente hay que suponer que es similar. Después de definir el espacio, se define una métrica en este espacio. Normalmente es la métrica de similitud de objetos, definida por la similitud de coseno [2], [4]. La idea de dicha similitud es: más se parecen los dos objetos entre sí, menor es el ángulo entre ellos en el espacio definido, y por lo tanto mayor es el coseno de este ángulo.

Ahora bien, la siguiente pregunta es: cómo elegir las características para definir el espacio vectorial. En este momento empiezan a prevalecer las consideraciones lingüísticas: es precisamente la parte lingüística que determina las características que podemos elegir. Por ejemplo, la idea más sencilla utilizada en recuperación de información es utilizar todas las palabras en varios documentos como sus características, y después comparar esos documentos: cuantas más palabras "iguales" tienen dos documentos, más se parecen entre sí estos documentos. Así se construye el espacio vectorial para la tarea de recuperación de información.

Obviamente en este caso, las ideas de qué tipo de información lingüística se puede utilizar se restringen por el requisito formal de utilizar el modelo de espacio vectorial, es decir, la obligación de representar los objetos como los conjuntos de valores de características.

La siguiente posibilidad más utilizada en la práctica, y que ya tiene cierto fundamento lingüístico, es la idea de utilizar n-gramas como características en el modelo de espacio vectorial. El concepto de n-grama es realmente muy sencillo: son las secuencias de palabras (u otros tipos de elementos) según aparecen en los textos. Entonces, el fundamento es que se tome en cuenta la información sintagmática de las palabras.

Sin embargo, sería muy provechoso utilizar conocimiento aún más "lingüístico", es decir, que involucre más información propiamente lingüística, por ejemplo, como en [5], [6], [7]. Como un camino en esta dirección, en nuestros trabajos anteriores [4], [8], [9], [10] hemos propuesto un nuevo concepto de n-gramas, que contiene aún una mayor información de naturaleza lingüística que los n-gramas tradicionales: n-gramas sintácticos. La idea de los n-gramas sintácticos consiste en construirlos siguiendo la ruta en el árbol sintáctico. De esa manera los n-gramas sintácticos siguen siendo n-gramas, pero permiten introducir información sintáctica [11] en los métodos de aprendizaje automático.

Sin embargo, en todos esos trabajos hemos propuesto obtener un n-grama sintáctico como un fragmento de una ruta continua, no se permiten bifurcaciones en la ruta —más adelante presentaremos ejemplos de eso. En este artículo vamos a presentar el concepto de n-gramas sintácticos no-continuos, es decir, siguiendo la ruta en el árbol sintáctico, se permite entrar en las bifurcaciones y regresar.

La estructura del resto del artículo es la siguiente. Primero discutimos el concepto de los n-gramas sintácticos continuos, tal como se presentó en nuestros trabajos anteriores. Después presentamos el concepto de los n-gramas sintácticos no-continuos. En las siguientes secciones vamos a considerar los ejemplos de extracción de los n-gramas sintácticos continuos y no-continuos para el español y el inglés. Finalmente, presentaremos las conclusiones.

II. N-GRAMAS SINTÁCTICOS CONTINUOS

En nuestros trabajos anteriores [4], [8], [9], [10] hemos introducido el nuevo concepto de los n-gramas sintácticos, es decir, n-gramas obtenidos siguiendo las rutas en un árbol

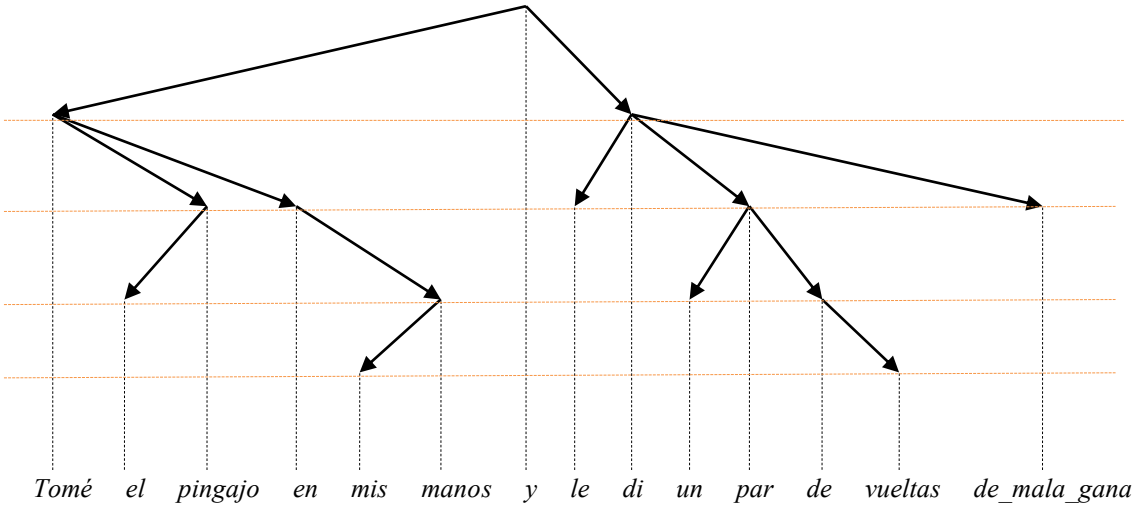

Fig. 1. Árbol sintáctico (analizado por FreeLing) representado con dependencias

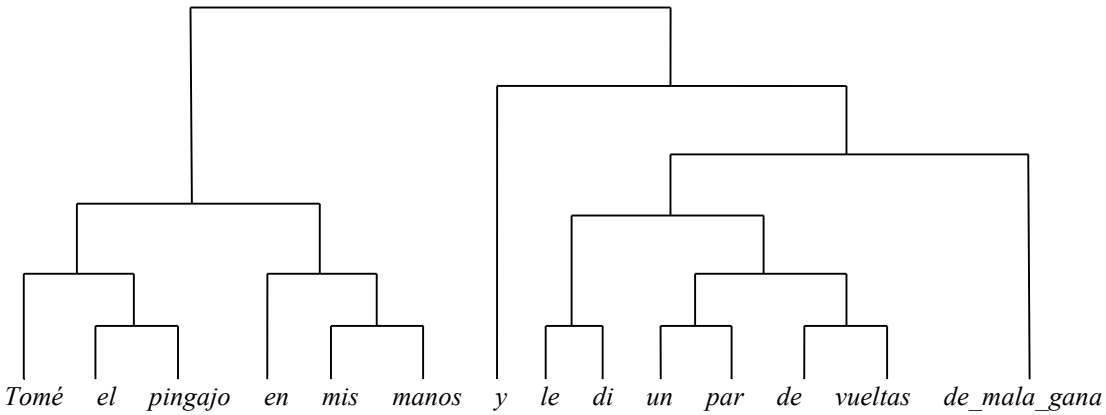

Fig. 2. Árbol sintáctico (analizado por FreeLing) representado con constituyentes

sintáctico. Un árbol sintáctico de una frase muestra[1] se presenta en las fig. 1 y 2, utilizando el formalismo de dependencias y constituyentes. Cabe mencionar que la idea de utilizar información estructural de relaciones de palabras en tareas específicas se ha presentado anteriormente [12], [13], [14], [15], [16], sin embargo, en ninguno de estos trabajos se ha generalizado, ni se ha relacionado con la idea de los n-gramas.

El trabajo [17] ha propuesto una idea similar en el campo de análisis semántico, donde la utilidad de la información sintagmática se demuestra en tareas muy específicas de: 1) "*semantic priming*", es decir, los experimentos psicolingüísticos sobre la similitud de palabras, 2) detección de sinónimos en las pruebas TOEFL, y 3) ordenamiento de sentidos de palabras según su importancia.

En nuestra opinión, este trabajo no tuvo mucha resonancia en otras tareas de PLN precisamente por no relacionar la información sintáctica con los n-gramas, que es la herramienta principal en la gran mayoría de tareas, ni por mostrar su utilidad en otras tareas que no son tan específicamente orientadas a la semántica.

Anteriormente, hemos propuesto varios tipos de n-gramas sintácticos con base en qué elementos los constituyen:

- Elementos léxicos (palabras, lemas, o raíces),
- Etiquetas de las categorías gramaticales (*POS tags*),
- Nombres de las relaciones sintácticas (*SR tags*),
- Caracteres,
- N-gramas sintácticos mixtos (combinaciones de los tipos anteriores).

Cabe mencionar que para la construcción de los n-gramas sintácticos de caracteres es necesario primero construir los n-gramas sintácticos de palabras y después basándose en éstos construir los n-gramas de caracteres. Es cuestión de trabajo futuro verificar si los n-gramas sintácticos de caracteres son útiles para algunas tareas de PLN. Resulta que la aplicación de

[1]Frase de una de las obras de A. Conan-Doyle.

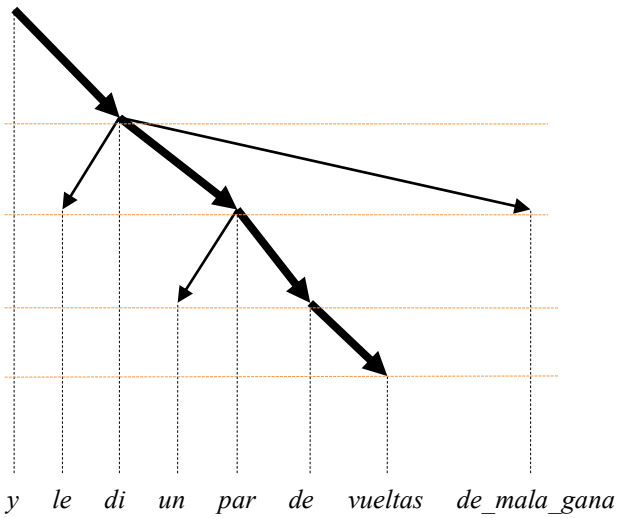

Fig. 3. N-gramas sintácticos continuos en el fragmento del árbol sintáctico: ejemplo de un 5-grama

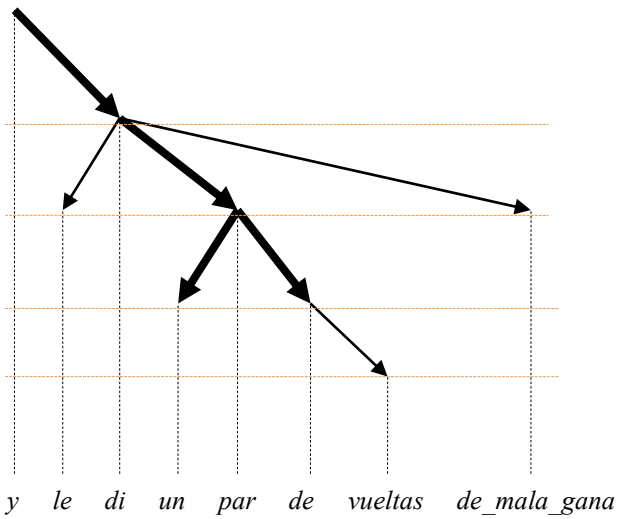

Fig. 4. N-gramas sintácticos no-continuos en el fragmento del árbol sintáctico: ejemplo de un 5-grama

los n-gramas tradicionales de caracteres presenta buenos resultados en algunas tareas, por ejemplo, en la tarea de detección de autoría [17]. Sin embargo, desde nuestro punto de vista, la aplicación de los n-gramas de caracteres es hasta cierto punto anti-intuitivo, y falta analizar las razones de su funcionamiento aceptable.

Respecto a los n-gramas mixtos, se deberá analizar a futuro qué combinaciones de los elementos: palabras, *POS-tags*, *SR-tags*, y en qué posiciones: al inicio, en medio de un n-grama, o al final, está dando mejores resultados.

En [17] se menciona la idea de ponderar las relaciones entre los elementos de un n-grama sintáctico. Esa idea no nos parece aplicable directamente en el contexto de modelos de espacio vectorial, donde los n-gramas son las características (dimensiones del espacio). Sin embargo, esa idea puede ser útil en el momento de calcular los pesos de n-gramas sintácticos, aparte de los valores tradicionales de *tf-idf*.

En nuestros trabajos anteriores [18] hemos demostrado que los n-gramas sintácticos pueden dar mejores resultados que los n-gramas tradicionales. Lo analizamos para el problema de detección de autoría. Sin embargo, los n-gramas sintácticos pueden ser utilizados en cualquier tipo de problemas donde se utilizan los n-gramas tradicionales, porque igual permiten la construcción del modelo de espacio vectorial. La desventaja de los n-gramas sintácticos consiste en el hecho que se requiere el análisis sintáctico automático previo, lo que toma cierto tiempo de procesamiento, aunque no es una limitación muy seria. También no para todos los idiomas existen analizadores sintácticos automáticos, pero sí para los idiomas con mayor presencia en el mundo, como el español o el inglés.

Independientemente del tipo de elementos que constituyen los n-gramas sintácticos, todos los n-gramas sintácticos considerados en nuestros trabajos anteriores, son continuos. Eso quiere decir que la ruta sintáctica que estamos siguiendo nunca tiene bifurcaciones, véase la comparación de la fig. 3 y

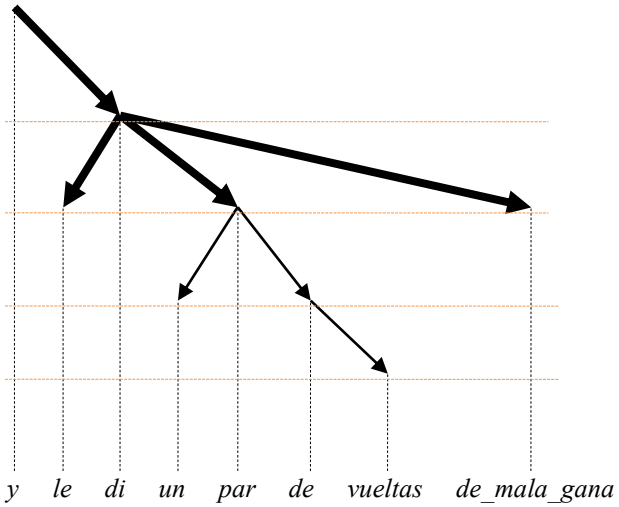

Fig. 5. N-gramas sintácticos no-continuos en el fragmento del árbol sintáctico: otro ejemplo de un 5-grama

de las fig. 4 y 5. La ruta marcada con flechas en negrita en la fig. 3 corresponde a un 5-grama sintáctico continuo. En este caso es: "*y di par de vueltas*".

A continuación presentaremos otro tipo de n-gramas sintácticos, donde se permiten las bifurcaciones.

III. N-GRAMAS SINTÁCTICOS NO-CONTINUOS

En esta sección vamos a presentar el complemento (o se puede considerar como una generalización) del concepto de n-gramas sintácticos continuos: n-gramas sintácticos no-continuos.

Como se puede observar en la sección anterior, la intuición detrás del concepto de n-gramas sintácticos continuos está relacionada principalmente con el hecho de que una secuencia

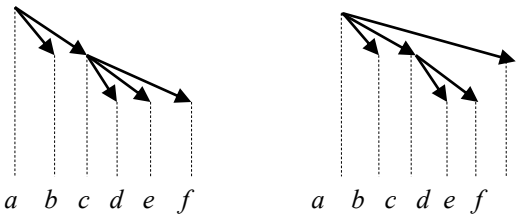

Fig. 6. Ejemplo de la ambigüedad en bifurcaciones

de palabras relacionadas puede ser considerada como tal, en su totalidad.

Sin embargo, existen otros conceptos lingüísticos interesantes, que no caben en el modelo de una secuencia unidimensional, por ejemplo, las valencias verbales (o patrones de rección) [19], [20]. Por ejemplo, el verbo *comprar* tiene los actantes: *quién, qué, de quién, por cuanto dinero*. Sería muy interesante tenerlos presentes al mismo tiempo en un n-grama. Sin embargo, tanto para el caso de n-gramas tradicionales, como de n-gramas sintácticos, todos esos componentes hubieran sido separados en n-gramas diferentes. Entonces, la intuición detrás del concepto de n-gramas sintácticos no-continuos es precisamente tratar de unir las palabras relacionadas semánticamente, aunque éstas no tengan una ruta continua, pero sí tengan alguna ruta que las conecte.

Es muy fácil dar una definición formal de n-gramas sintácticos no-continuos: son todos los sub-árboles de longitud *n* de un árbol sintáctico. En este sentido, digamos, los n-gramas sintácticos continuos se definen como todos los sub-árboles sin bifurcaciones de longitud *n* de un árbol sintáctico. Eso quiere decir que los n-gramas sintácticos continuos son un caso particular de n-gramas sintácticos no-continuos, al aplicar la definición propuesta. La longitud de un árbol es el número de arcos en este árbol, lo que corresponde al valor de *n* (en caso de los n-gramas).

Otro término que podemos proponer para denotar los n-gramas sintácticos no-continuos, es t-n-gramas (*tree n-grams*, en inglés) es decir, n-gramas de árboles[2].

Es cuestión de un trabajo futuro determinar qué tipo de n-gramas: continuos o no-continuos, son mejores para varios tipos de tareas de la lingüística computacional. Es posible que para algunos tipos de tareas son mejores unos, y para otros tipos de tareas, otros.

Cabe mencionar que el número de n-gramas sintácticos no-continuos es mayor que el de los n-gramas sintácticos continuos, dado que los últimos son un caso particular de los primeros.

El algoritmo de construcción (u obtención) de los n-gramas sintácticos no-continuos es relativamente sencillo. Para el nodo raíz hay que considerar todas las posibles combinaciones de sus hijos con el tamaño no mayor que *n*, y repetir este procedimiento de manera recursiva para cada nodo hijo. Y así de manera sucesiva hay que pasar por todos los nodos del árbol sintáctico.

Puede surgir la pregunta ¿y cómo representar a los n-gramas sintácticos no-continuos sin utilizar su representación gráfica? Cabe recordar que los n-gramas sintácticos continuos son simplemente secuencias de palabras, pero el caso de los n-gramas sintácticos no-continuos es diferente. Estamos proponiendo utilizar los siguientes convenios. Nótese que son convenios, por lo que pueden ser modificados en un futuro. Dentro de cada n-grama sintáctico no-continuo pueden existir unas partes continuas y una o varias bifurcaciones. Vamos a separar los elementos continuos de n-gramas con espacios en blanco —nada más, y en la parte de la bifurcación vamos a poner comas, además vamos a usar paréntesis para la parte de bifurcaciones, dado que posteriormente puede aparecer la ambigüedad de estructuras.

Dos ejemplos de los 5-gramas sintácticos no-continuos están representados en la fig. 4 y la fig. 5: "*y di par [un, de]*", "*y di [le, par, de_mala_gana]*". Nótese que los paréntesis y las comas ahora son partes de los n-gramas, pero eso de ninguna manera impide la identificación de los n-gramas sintácticos que son iguales.

La ambigüedad puede aparecer por ejemplo, en caso de que un n-grama tenga dos bifurcaciones y varios fragmentos continuos. Por ejemplo, el n-grama "*a [b, c [d, e, f]]*" y "*a [b, c [d, e], f]*" tiene el nodo *f* o bien como el tercer nodo debajo del nodo *c*, o bien como el tercer nodo debajo del nodo *a,* véase la fig. 6.

Ahora bien, tenemos dos posibilidades de manejar las partes con bifurcaciones: 1) tal como aparecen en el texto, que es la manera más natural, o 2) ordenarlos de alguna manera, por ejemplo, alfabéticamente. Esta última opción nos permite tomar en cuenta los cambios relacionados con el orden de palabras. Sin embargo, se necesitan más investigaciones para determinar cuál de las dos opciones es mejor y para qué tarea de PLN.

Otra posibilidad que nos gustaría mencionar es marcar directamente dentro de los n-gramas sintácticos no-continuos su profundidad. La intuición detrás de esta idea es que para algunos tipos de n-gramas sintácticos no-continuos puede ser importante su posición en el árbol sintáctico de la oración. En este caso la notación sería: "y_1 *di*$_2$ *par*$_3$ *[un*$_4$*, de*$_4$*]*", "y_1 *di*$_2$ *[le*$_3$*, par*$_3$*, de_mala_gana*$_3$*]*". Técnicamente, sería suficiente marcar el nivel de la primera palabra únicamente; podemos marcar los demás niveles también, pero no es estrictamente necesario.

A continuación vamos a considerar dos ejemplos de construcción de n-gramas sintácticos no-continuos, uno para el español y otro para el inglés, y los comparamos con los n-gramas sintácticos continuos.

[2] La sugerencia de A. Gelbukh es usar el término "t-gramas, árbol-gramas" (*tree grams*, *t-grams*, en inglés), sin embargo nos parece un poco mejor el término "n-gramas de árboles" (*tree n-grams*, *t-n-grams,* en inglés), dado que así se establece la relación del término propuesto con el concepto muy tradicional de los n-gramas. Del otro lado, una consideración a favor del término "t-grama" es su forma más simple.

IV. Ejemplo de construcción de n-gramas sintácticos no-continuos para el español

En esta sección vamos a presentar ejemplos de la construcción de n-gramas sintácticos continuos y no-continuos para el español. Tomemos la frase muestra:

Tomé el pingajo en mis manos y le di un par de vueltas de mala gana.

Para construir automáticamente los n-gramas sintácticos, es necesario aplicar antes un programa de análisis sintáctico automático, llamado en inglés *parser*. Para el español hemos utilizado el programa FreeLing [22], [23], que está disponible de manera gratuita.

El analizador sintáctico puede construir el árbol utilizando dos formatos: constituyentes y dependencias. El árbol de dependencias se presenta en la fig. 1, y el de constituyentes en la fig. 2. Ambos formatos tienen esencialmente la misma información de relaciones de palabras. Para los fines de construcción de los n-gramas sintácticos nos parece mejor utilizar las dependencias, porque la representación es más transparente. Sin embargo, de igual manera se puede utilizar el árbol de constituyentes.

Cabe mencionar que el analizador sintáctico primero realiza el análisis morfológico y la lematización. Como se puede observar, a cada palabra de la oración le corresponde su lema y la información gramatical, por ejemplo, "*Tomé tomar VMIS1S0*". Primero viene la palabra, después el lema, y al final la información gramatical.

La información gramatical utiliza el esquema de codificación EAGLES, que es el estándar de facto para el análisis morfológico automático del español. Por ejemplo, en la etiqueta VMIS1S0, la primera letra "V" corresponde al "verbo" ("N" hubiera sido un sustantivo, "A" un adjetivo, etc.), la "I" es el indicativo, la "S" significa el pasado, "1" es la primera persona, la otra letra "S" significa "singular". Como se puede observar, en cada posición se codifica un tipo específico de la información gramatical, y cada etiqueta tiene como máximo siete posiciones, de las cuales algunos pueden no usarse en algunos casos, por ejemplo, en caso de los sustantivos.

Primero presentamos los resultados de análisis automático de la oración utilizando el formalismo de constituyentes (fig. 2).

```
+coor-vb_[
 grup-verb_[
  +verb_[
   +(Tomé tomar VMIS1S0 -)
  ]
  sn_[
   espec-ms_[
    +j-ms_[
     +(el el DA0MS0 -)
    ]
   ]
   +grup-nom-ms_[
    +n-ms_[
     +(pingajo pingajo NCMS000 -)
    ]
   ]
  ]
  grup-sp_[
   +prep_[
    +(en en SPS00 -)
   ]
   sn_[
    espec-fp_[
     +pos-fp_[
      +(mis mi DP1CPS -)
     ]
    ]
    +grup-nom-fp_[
     +n-fp_[
      +(manos mano NCFP000 -)
     ]
    ]
   ]
  ]
]
+(y y CC -)
grup-verb_[
 patons_[
  +paton-s_[
   +(le le PP3CSD00 -)
  ]
 ]
 +grup-verb_[
  +verb_[
   +(di dar VMIS1S0 -)
  ]
 ]
 sn_[
  espec-ms_[
   +indef-ms_[
    +(un uno DI0MS0 -)
   ]
  ]
  +grup-nom-ms_[
   +n-ms_[
    +(par par NCMS000 -)
   ]
  ]
  sp-de_[
   +(de de SPS00 -)
   sn_[
    +grup-nom-fp_[
     +n-fp_[
      +(vueltas vuelta NCFP000 -)
     ]
    ]
   ]
  ]
 ]
 sadv_[
  +(de_mala_gana de_mala_gana RG -)
 ]
```

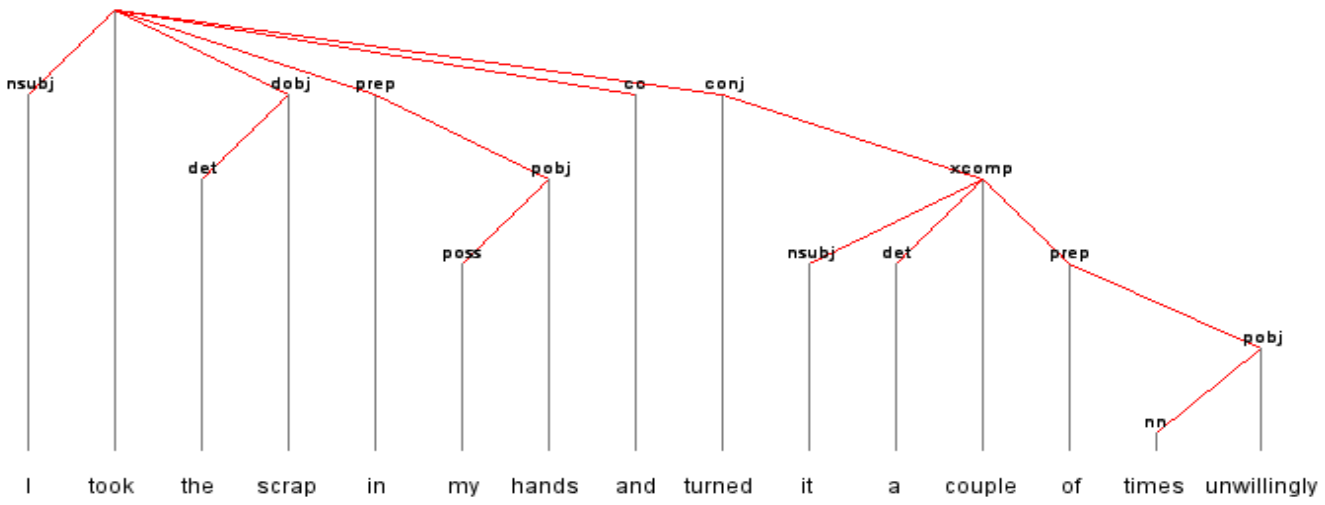

Fig. 7. El árbol sintáctico del ejemplo para el inglés

```
 ]
 F-term_[
  +(. . Fp -)
 ]
]
```

Información muy similar se presenta utilizando el formalismo de dependencias en la fig. 1.

```
coor-vb/top/(y y CC -) [
 grup-verb/co-v/(Tomé tomar VMIS1S0 -) [
  sn/dobj/(pingajo pingajo NCMS000 -) [
   espec-ms/espec/(el el DA0MS0 -)
  ]
  grup-sp/sp-obj/(en en SPS00 -) [
   sn/obj-prep/(manos mano NCFP000 -) [
    espec-fp/espec/(mis mi DP1CPS -)
   ]
  ]
 ]
 grup-verb/co-v/(di dar VMIS1S0 -) [
  patons/iobj/(le le PP3CSD00 -)
  sn/dobj/(par par NCMS000 -) [
   espec-ms/espec/(un uno DI0MS0 -)
   sp-de/sp-mod/(de de SPS00 -) [
    sn/obj-prep/(vueltas vuelta NCFP000 -)
   ]
  ]
  sadv/cc/(de_mala_gana de_mala_gana RG -)
 ]
 F-term/modnomatch/(. . Fp -)
]
```

Como ya mencionamos es más sencillo utilizar las dependencias, porque ellas prácticamente ya contienen los n-gramas sintácticos.

Se puede observar que las tres palabras "*de_mala_gana*" realmente son un solo adverbio.

Ahora bien, vamos a presentar los ejemplos de los n-gramas sintácticos extraídos. Primero presentamos los n-gramas sintácticos continuos.

Los bi-gramas sintácticos (en bi-gramas no hay diferencia entre bi-gramas continuos y no-continuos) son:

y tomé, tomé pingajo, pingajo el, tomé en, en manos, manos mis, y di, di le, di par, par un, par de, de vueltas, di de_mala_gana.

Los tri-gramas continuos son:

y tomé pingajo, y tomé en, tomé pingajo el, tomé en manos, en manos mis, y di le, y di par, y di de_mala_gana, di par un, di par de, par de vueltas.

Los 4-gramas continuos son:

y tomé pingajo el, y tomé en manos, tomé en manos mis, y di par un, y di par de, di par de vueltas.

No vamos a repetir los mismos elementos —elementos continuos—, aunque ellos también formen parte de los n-gramas sintácticos no-continuos. Nótese que en este caso tenemos que usar la notación propuesta para los n-gramas no-continuos para poder distinguirlos de otras configuraciones posibles. La notación forma parte de los n-gramas, no es algo adicional, es el propio n-grama. En los n-gramas sintácticos continuos las comas no son parte de n-grama, se pueden omitir poniendo un n-grama por línea. Entonces, como la coma ya es una parte de la notación, para ser más claros, ahora sí vamos a presentar un n-grama por línea. Entonces, los tri-gramas no-continuos nuevos en comparación con los n-gramas no-continuos son:

tomé [pingajo en]
di [le par]
di [le de_mala_gana]
di [par de_mala_gana]

par [un de]

Los 4-gramas no-continuos nuevos son:

tomé [pingajo el, en]
tomé [pingajo, en manos]
di [le, par un]
di [le, par de]
di [le, par, de_mala_gana]
di [par un, de_mala_gana]
di [par de, de_mala_gana]
par [un, de vueltas]

V. Ejemplo de construcción de n-gramas sintácticos no-continuos para el inglés

En esta sección vamos a analizar la construcción de n-gramas sintácticos para el inglés. Para simplificar la comparación con el español, tomaremos la traducción de la misma frase que en la sección anterior, la fig. 7, —en este caso la figura se generó de manera automática.

I took the scrap in my hands and turned it a couple of times unwillingly.

Si vamos utilizar el mismo analizador sintáctico, es decir, FreeLing, los resultados van a ser muy similares. Vamos a probar con otro analizador sintáctico para el inglés que también es bien conocido —analizador sintáctico de Stanford (Stanford *parser*) [24]. El árbol de constituyentes es como sigue:

```
(ROOT
 (S
  (NP (PRP I))
  (VP
   (VP (VBD took)
    (NP (DT the) (NN scrap))
    (PP (IN in)
     (NP (PRP$ my) (NNS hands))))
   (CC and)
   (VP (VBD turned)
    (S
     (NP (PRP it))
     (NP
      (NP (DT a) (NN couple))
      (PP (IN of)
       (NP (NNS times) (NN unwillingly)))))))
  (. .)))
```

En este *parser*, para el árbol de dependencias se usa una representación muy simple, pero expresiva: nombre de la relación, dos palabras (o sus *POS tags*, o lemas) junto con sus números en la oración. Primero se menciona la palabra principal y después la dependiente, es decir, el orden de las palabras es importante. Esta información permite construir el árbol sintáctico de manera única.

```
nsubj(took-2, I-1)
root(ROOT-0, took-2)
det(scrap-4, the-3)
dobj(took-2, scrap-4)
prep(took-2, in-5)
poss(hands-7, my-6)
pobj(in-5, hands-7)
cc(took-2, and-8)
conj(took-2, turned-9)
nsubj(couple-12, it-10)
det(couple-12, a-11)
xcomp(turned-9, couple-12)
prep(couple-12, of-13)
nn(unwillingly-15, times-14)
pobj(of-13, unwillingly-15)
```

Cabe mencionar que hemos desarrollado un programa en el lenguaje de programación *Python* que convierte el formato de dependencias de FreeLing al formato de dependencias de Stanford *parser*. Un problema relacionado con esa conversión consiste en que el formato de FreeLing no contiene los números de palabras en la oración, por lo que no se puede reconstruir el árbol de entrada en todos los casos, pero eso no afecta la construcción de los n-gramas sintácticos: se asigna un número consecutivo para identificar las palabras.

Se puede observar que aunque la frase es muy parecida, el otro *parser* aplicó otras reglas, específicamente, manejó de manera diferente la conjunción y cometió varios errores: por ejemplo, con *unwillingly*, relacionándolo con *of* y no con *turned;* con *it,* relacionándolo con *couple* y no con *turned*. Sin embargo, eso no afecta de manera conceptual nuestra discusión, ya que nuestra tarea no es mejorar algún *parser*. Ahora procedamos a la construcción de los n-gramas sintácticos continuos y no-continuos.

Los bi-gramas sintácticos (recordemos que no hay diferencia entre los bi-gramas sintácticos continuos y no-continuos) son:

took I, took scrap, scrap the, took in, in hands, hands my, took and, took turned, turned couple, couple it, couple a, couple of, of unwillingly, unwillingly times.

Los tri-gramas sintácticos continuos son:

took scrap the, took in hands, in hands my, took turned couple, turned couple it, turned couple a, turned couple of, couple of unwillingly, of unwillingly times.

Los 4-gramas sintácticos continuos son:

took in hands my, took turned couple it, took turned couple a, took turned couple of, turned couple of unwillingly, couple of unwillingly times.

Como en el ejemplo anterior no vamos a repetir los mismos elementos, aunque los n-gramas continuos forman parte de los n-gramas sintácticos no-continuos.

Los tri-gramas no-continuos nuevos son (puede ser que algunos de ellos son errores del *parser*, eso no afecta la idea propuesta dado que son errores de otro tipo, que se puede corregir mejorando el propio *parser*):

took [I, scrap]
took [I, in]
took [I, and]
took [I, turned]
took [scrap, in]
took [scrap, and]
took [scrap, turned]
took [in, and]
took [in, turned]
took [and, turned]
couple [it, a]
couple [it, of]
couple [a, of]

Los 4-gramas no-continuos nuevos son:

took [I, scrap the]
took [in, scrap the]
took [and, scrap the]
took [turned, scrap the]
took [I, in hands]
took [scrap, in hands]
took [and, in hands]
took [turned, in hands]
took [I, scrap, in]
took [I, scrap, and]
took [I, scrap, turned]
took [scrap, in, and]
took [scrap, in, turned]
took [in, and, turned]
couple[it, a, of]
couple[it, of unwillingly]
couple[a, of unwillingly]

Nótese que en este caso hemos tomado los elementos de los n-gramas sintácticos no-continuos en su orden de aparición en el texto. Como mencionamos, otra opción es ordenarlos de alguna manera, por ejemplo, alfabéticamente.

VI. Conclusiones

En este artículo hemos discutido cómo se realiza la investigación en la etapa moderna de la lingüística computacional, sobre todo relacionada con el uso de los métodos de aprendizaje automático [25], y hemos presentado una idea novedosa de n-gramas sintácticos no-continuos como posibles características en un modelo de espacio vectorial. Lo hemos comparado con nuestra idea presentada anteriormente de n-gramas sintácticos continuos.

Hemos considerado dos ejemplos para el español e inglés, brevemente hemos presentado el algoritmo de construcción de los n-gramas sintácticos no-continuos y hemos propuesto la notación formal de su representación. La notación —las comas y los paréntesis— es importante porque sin ella no hay manera de mantener la estructura —en principio multidimensional— de los n-gramas sintácticos no-continuos.

Se necesitan múltiples estudios experimentales para determinar qué parámetros de construcción de los n-gramas sintácticos no-continuos son mejores y para qué tipo de tareas existentes dentro de la lingüística computacional.

Agradecimientos

Trabajo realizado con el apoyo de gobierno de la Ciudad de México (proyecto ICYT-DF PICCO10-120), el apoyo parcial del gobierno de México (CONACYT, SNI) e Instituto Politécnico Nacional, México (proyectos SIP 20120418, 20131441, 20131702; COFAA), proyecto FP7-PEOPLE-2010-IRSES: Web Information Quality - Evaluation Initiative (WIQ-EI) European Commission project 269180. Agradezco a A. Gelbukh por su ayuda y sugerencias fructíferas.

Referencias

[1] I.A. Bolshakov, A. Gelbukh. *Computational linguistics: models, resources, applications*. 187 pp, 2004.

[2] C. Manning, H. Schütze, "Foundations of Statistical Natural Language Processing," MIT Press, Cambridge, MA, 1999.

[3] A. Gelbukh, G. Sidorov, "Procesamiento automático del español con enfoque en recursos léxicos grandes," IPN, 307 p., 2010.

[4] G. Sidorov, "N-gramas sintácticos y su uso en la lingüística computacional," *Vectores de investigación*, 6(6), 15 p., 2013.

[5] J.A. Reyes, A. Montes, J.G. González, D.E. Pinto, "Clasificación de roles semánticos usando características sintácticas, semánticas y contextuales," *Computación y sistemas*, 17(2), pp. 263–272, 2013.

[6] A. Gelbukh. "Using a semantic network for lexical and syntactical disambiguation," *Proc. CIC-97, Simposium Internacional de Computación*, Mexico, pp. 352–366, 1997.

[7] Y. Ledeneva, A. Gelbukh, R.A. García-Hernández. "Terms Derived from Frequent Sequences for Extractive Text Summarization," *Proc. CICLing 2008, Lecture Notes in Computer Science*, N 4919, pp. 593–604, 2008.

[8] G. Sidorov, F. Velasquez, E. Stamatatos, A. Gelbukh, L. Chanona-Hernández, "Syntactic Dependency-based N-grams as Classification Features," *LNAI*, 7630, pp. 1–11, 2012.

[9] G. Sidorov, F. Velasquez, E. Stamatatos, A. Gelbukh, L. Chanona-Hernández, "Syntactic Dependency-Based N-grams: More Evidence of Usefulness in Classification," *LNCS*, 7816 (Proc. of CICLing), pp. 13–24 (2013)

[10] G. Sidorov, F. Velasquez, E. Stamatatos, A. Gelbukh, L. Chanona-Hernández, "Syntactic N-grams as Machine Learning Features for Natural Language Processing," *Expert Systems with Applications*, in press, 8 p., 2013.

[11] H. Calvo, J.O. Juarez Gambino, A. Gelbukh, K. Inui. "Dependency Syntax Analysis using Grammar Induction and a Lexical Categories Precedence System," *Proc. of CICLing 2011, Lecture Notes in Computer Science*, N 6608, pp. 109–120 , 2011.

[12] M. Khalilov, J.A.R. Fonollosa, "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*, pp. 424–432, 2009.

[13] N. Habash, "The Use of a Structural N-gram Language Model in Generation-Heavy Hybrid Machine Translation," *LNCS*, 3123, pp. 61–69, 2004.

[14] A. Agarwal, F. Biads, K.R. McKeown, "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), pp. 24–32, 2009.

[15] A. Gelbukh. "Syntactic disambiguation with weighted extended subcategorization frames," *Proc. PACLING-99, Pacific Association for Computational Linguistics*, Waterloo, Canada, August 25–28, pp. 244–249, 1999.

[16] A. Gelbukh, I. Bolshakov, S. Galicia-Haro. "Statistics of parsing errors can help syntactic disambiguation," *Proc. CIC-98, Simposium Internacional de Computación*, November 11–13, Mexico City, pp. 405–515, 1998.

[17] S. Pado y M. Lapata, "Dependency-based construction of semantic space models," *Computational Linguistics*, 33(2), pp. 161–199, 2007.

[18] A. Gelbukh. "Natural language processing: Perspective of CIC-IPN," *International Conference on Advances in Computing, Communications and Informatics (ICACCI 2013)*, IEEE Conference Publications, pp. 2112–2121, 2013.

[19] S.N. Galicia-Haro, A. Gelbukh, I.A. Bolshakov. "Acquiring syntactic information for a government pattern dictionary from large text corpora," *IEEE International Workshop on Natural Language Processing and Knowledge Engineering, NLPKE 2001 at International IEEE SMC-2001 Conference: Systems, Man, and Cybernetics*. Tucson, USA, October 7–10, IEEE, pp. 536–542, 2001.

[20] A. Gelbukh, I. Bolshakov, S. Galicia Haro. "Automatic Learning of a Syntactical Government Patterns Dictionary from Web-Retrieved Texts," *Proc. International Conference on Automatic Learning and Discovery*, Carnegie Mellon University, Pittsburgh, PA, USA, June 11–13, pp. 261–267, 1998.

[21] M. Koppel, J. Schler, S. Argamon, "Authorship attribution in the wild," *Language Resources and Evaluation* 45(1), pp. 83–94, 2011.

[22] X. Carreras, I. Chao, L. Padró, M. Padró, "FreeLing: An Open-Source Suite of Language Analyzers," *Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC'04),* 2004.

[23] L. Padró, E. Stanilovsky, "FreeLing 3.0: Towards Wider Multilinguality," *Proceedings of the Language Resources and Evaluation Conference (LREC 2012),* ELRA, Turkey, 2012.

[24] M.C. de Marneffe, B. MacCartney, C.D. Manning, "Generating Typed Dependency Parses from Phrase Structure Parses," *Proc. of LREC*, 2006.

[25] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, I.H. Witten, "The WEKA Data Mining Software: An Update," *SIGKDD Explorations*, 11(1), 2009.