POS Tagger basado en HMM y SVM

Javier Andrés Redolfi
CIII - Centro de Investigación en Informática para la Ingeniería
Universidad Tecnológica Nacional - Facultad Regional Córdoba
Córdoba, Argentina

19 de diciembre de 2013

Índice

1. Introducción

En este informe se presenta la construcción de un part-of-speech (POS) tagger. Un POS Tagger se encarga de asignar (o etiquetar) a cada una de las palabras de un texto con su categoría gramatical.

1.1. Objetivos

El POS Tagger debe tener las siguientes características:

- especializado para preguntas
- proporcionar una salida probabilística
- debe permitir que la entrada sean los POS tags

1.2. Asunciones

Se parten de las siguientes hipótesis:

- alto grado de correlación entre un POS y sus vecinos anteriores y posteriores
- fuerte estructura de las oraciones a procesar, en este caso preguntas
  - empiezan y terminan con signos de interrogación
  - en las primeras palabras normalmente tenemos pronombres interrogativos

2. Desarrollo

2.1. Solución propuesta

En la literatura existen muchos esquemas que abordan el problema de POS tagging, los cuales difieren entre sí en el método de aprendizaje y en la complejidad del modelo construido. Los formalismos comúnmente utilizados son Modelos de Markov (n-gramas), reglas de transformación, árboles de decisión, redes neuronales [6], autómatas y transductores de estados finitos, etc. En este trabajo se optó por el uso de Modelos Ocultos de Markov [12] (HMM, por sus siglas en inglés), los cuales ya han sido usado en este tipo de tareas [1].

2.1.1. Modelos Ocultos de Markov

Un modelo oculto de Markov es un modelo estadístico en el que se asume que el sistema a modelar es un proceso de Markov de parámetros desconocidos. El objetivo es determinar los parámetros desconocidos (u ocultos, de ahí el nombre) de dicha cadena a partir de los parámetros observables. Los parámetros extraídos se pueden emplear para llevar a cabo sucesivos análisis, por ejemplo en aplicaciones de reconocimiento de patrones. Dentro de las aplicaciones más comunes en reconocimiento temporal de patrones podemos nombrar reconocimiento del habla, de dígitos escritos a mano, de gestos, POS tagging, partituras musicales, etc.

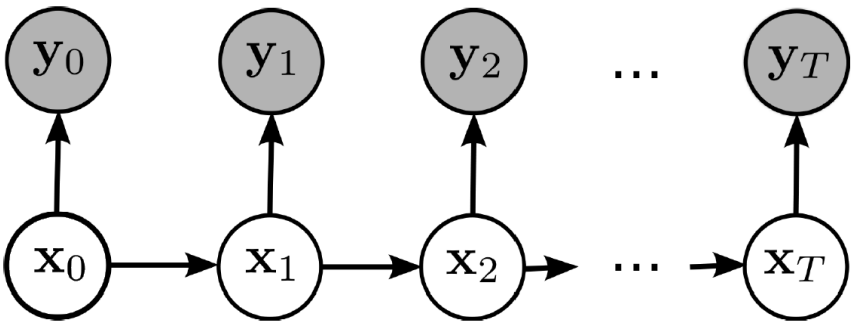

Figura 1: Modelo Oculto de Markov

2.1.2. Justificación de su Uso

La elección de HMM para la resolución del problema planteado se basó en que este modelo se ajusta a los objetivos e hipótesis planteadas.
Por el lado de las hipótesis, los HMM nos permiten modelar el estado inicial más probable que para nuestro caso normalmente es un pronombre interrogativo; también nos permiten modelar las transiciones entre los POS tags, las cuales suponemos que tienen una alta correlación. Otra característica que tienen los HMM es que nos permiten modelar la generación de features dado el POS tag, esto nos da la posibilidad de probar con distintas features, agregando y sacando hasta encontrar las más óptimas.
Con respecto a los objetivos, al necesitar entrenamiento, lo podemos especializar para preguntas. Además como es un modelo probabilístico, su salida es un valor de probabilidad. La posibilidad de jugar con distintas features, nos permite que la salida (POS tags) sea también la entrada, estos POS tags de entrada deben ser obtenidos con otro POS tagger.

2.2. Diseño del HMM

En la figura 1 se puede ver un diagrama del modelo utilizado, en donde los x son los estados (POS tags en nuestro caso) y las y son las observaciones o features. Las flechas horizontales representan las probabilidades de transición y las verticales las probabilidades de emisión.

2.2.1. Probabilidades de Transición

Las probabilidades de transición son las probabilidades de pasar de un estado a otro. Para estimar estas probabilidades partimos de un corpus de preguntas anotado y contamos la cantidad de veces que se produce una transición de un estado x_i a otro estado x_j para todos los posibles estados. Luego estas cuentas son normalizadas para que representen una probabilidad.

2.2.2. Probabilidades de Emisión

Estas probabilidades modelan la probabilidad de que se genere una determinada observación dado el estado o POS tag. Para cada POS tag necesitamos un modelo diferente (igual modelo con distintos parámetros). El modelo elegido en este trabajo son las Máquinas de Soporte Vectorial [7] (SVM, por sus siglas en inglés) con kernel lineal.

Las SVM son muy utilizadas en problemas de clasificación y regresión. Dado un conjunto de ejemplos de entrenamiento (de muestras) etiquetadas con su clase correspondiente, podemos entrenar una SVM para construir un modelo que prediga la clase de una nueva muestra. Intuitivamente, una SVM es un modelo que representa a los puntos de muestra en el espacio, separando las clases por un margen lo más amplio posible. Cuando las nuevas muestras se ponen en correspondencia con dicho modelo, en función de su proximidad pueden ser clasificadas a una u otra clase. En nuestro caso las clases son los POS tags y los ejemplos de entrenamiento son las observaciones del HMM. La salida de una SVM es un score proporcional a la distancia de la observación al hiperplano, este score no es una probabilidad por lo tanto lo debemos calibrar entre 0 y 1 para que represente una probabilidad. Esto se hace ajustando una función sigmoide, para más referencias ver [11].

2.3. Features

Las features son las observaciones del HMM, en nuestro diseño estas features se utilizan como ejemplos de entrenamiento de las SVM. En este trabajo se utilizaron las siguientes:

- POS tag, utilizando el POS tagger SENNA [5] (POS)
- es número, signo de puntuación, mayúsculas (REG, por regular expresión)
- sufijos de tamaño 1 (suf1)
- sufijos de tamaño 2, los más comunes (suf2)
- sufijos de tamaño 3, los más comunes (suf3)
- prefijos de tamaño 2, los más comunes (pre2)
- prefijos de tamaño 3, los más comunes (pre3)

Cada una las características anteriores generan vectores que luego son concatenados para formar un sólo vector. Entre paréntesis se indica el nombre que se usará en el resto del informe para nombrar a cada una de estas características. La concatenación se indica con el signo +, por ejemplo POS+REG indica que usamos como característica la concatenación del POS tag con las expresiones regulares. La dimensionalidad del vector generado concatenando todas las características anteriores es de 807.

3. Resultados

3.1. Configuración Experimental

3.1.1. Corpus

El corpus utilizado fue QuestionBank [9], el cual es un corpus de 4000 preguntas anotadas con su part-of-speech tag. Este fue dividido en 2 partes, 90 % para entrenamiento y 10 % para test.

3.1.2. Baseline

Para comparar el funcionamiento del POS tagger, se generaron 2 baselines usando el SENNA POS tagger [5] y el POS tagger basado en HMM de NLTK [3].

3.1.3. Herramientas Utilizadas

Se utilizaron las librerías de software NLTK [2], scikit-learn [10], libsvm [4] y senna [5]. El código fue implementado en python y algunos scripts de bash.

3.2. Análisis de Características

En esta subsección a través de algunos ejemplos se analiza como el agregado de ciertas características influye en el resultado del POS tagging.

3.2.1. Agregado de sufijos como características

En el siguiente ejemplo se muestra como el agregado de sufijos de tamaño 2 me ayuda a corregir ciertos errores.

pregunta	What	is	the	name	of	the	managing	director	of	...
ground truth	WP	VBZ	DT	NN	IN	DT	JJ	NN	IN	...
POS+REG+suf1	WP	VBZ	DT	NN	IN	DT	NN	NN	IN	...
POS+REG+suf2	WP	VBZ	DT	NN	IN	DT	JJ	NN	IN	...

3.2.2. Agregado de verbos auxiliares como características

En el siguiente ejemplo vemos como el agregado de verbos auxiliares mejora el resultado. Aquí lo vemos en forma indirecta, porque no agregamos los verbos auxiliares sino los sufijos de tamaño 2, en este caso **is** al tener tamaño 2 se agrega como sufijo.

pregunta	What	is	the	name	of	a	hotel	in	Indianapolis	...
ground truth	WP	VBZ	DT	NN	IN	DT	NN	IN	NNP	...
POS	WP	VBD	DT	NN	IN	DT	NN	IN	NNP	...
POS+REG+suf2	WP	VBZ	DT	NN	IN	DT	NN	IN	NNP	...

3.2.3. Preguntas terminadas en Punto

En el siguiente ejemplo vemos uno de los problemas de los HMM, en este caso como la mayoría de las preguntas de entrenamiento termina con **?**, el modelo predice este estado como estado final con mucha mayor probabilidad que los restantes.

pregunta	Name	a	stimulant	.
ground truth	VB	DT	NN	.
POS	VB	DT	NN	?

3.3. Precisión

En la tabla 1 se muestra la precisión (0 a 1) obtenida con el tagger diseñado, comparada para distintas combinaciones de features y también la comparación con respecto al baseline. Esta precisión se calculo sobre el conjunto reservado para test. Se muestran los valores de media y varianza para 10 corridas.

Tagger/Feature	media	desviación estándar
SENNA POS Tagger	0.83	0.0055
NLTK HMM POS Tagger	0.81	0.0041
POSTag	0.85	0.0039
POSTag + REG	0.87	0.0025
POSTag + REG + suf1	0.88	0.0039
POSTag + REG + suf1 + suf2	0.88	0.0038
POSTag + REG + suf1 + suf2 + suf3	0.88	0.0038
POSTag + REG + suf1 + suf2 + suf3 + pre2	0.88	0.0044
POSTag + REG + suf1 + suf2 + suf3 + pre2 + pre3	0.88	0.0052
REG + suf1 + suf2 + suf3 + pre2 + pre3	0.80	0.0071

Tabla 1: Comparación de la precisión usando diferentes features y POS taggers

nº componentes	precisión	% varianza
807	0.88	100
200	0.88	92
100	0.88	85
70	0.88	81
50	0.87	76
10	0.84	50

Tabla 2: Precisión y porcentaje de varianza usando PCA, variando el número de componentes principales

3.4. Selección de Características

En este apartado se analiza el impacto del uso de selección de características o feature selection. El proceso de selección de características nos permite tener una medida de la discriminabilidad de las características, lo cual puede ser útil para la reducción de la dimensionalidad del problema con la consecuente disminución de la complejidad y tiempo de computo.

3.4.1. Análisis de Componentes Principales

El análisis de componentes principales o PCA por sus siglas en inglés es una técnica utilizada para reducir la dimensionalidad de un conjunto de datos, proyectando estos datos a un subespacio de menor dimensión, pero con sus dimensiones ordenadas de manera decreciente con respecto a la varianza y eliminando las de menor varianza.

En la tabla 2 se muestra la precisión variando el número de componentes principales utilizado. Como se puede ver, los valores de precisión no aumentan significativamente más allá de las 50 componentes. La desventaja del análisis usando PCA es que en este nuevo subespacio las nuevas características no tienen un sentido interpretable, por esto no podemos saber cuales son las características más representativas y cuales no, cuales se pueden quitar o cuales podemos agregar.

nº componentes	precisión
807	0.88
200	0.88
100	0.88
70	0.87
50	0.87
10	0.76

Tabla 3: Precisión usando Árboles Aleatorios, variando el número de componentes seleccionadas

Otro análisis que se puede realizar cuando aplicamos PCA es el de la explicación de la varianza (explained variance), el cual compara la varianza de los datos en el nuevo subespacio con la varianza en el espacio original. Esto se puede ver en la tabla 2.

3.4.2. Basada en Árboles Aleatorios

El método de selección basado en árboles aleatorios [8] nos permite seleccionar el subconjunto de características más representativo de los datos. La medida de representatividad de una característica está basada en la información mutua o alternativamente en la ganancia de información al realizar una división en el árbol usando esa característica. La ventaja de este método con respecto a PCA, es que al usar árboles su resultado es interpretable.

En la tabla 3 se puede ver la precisión del POS tagger usando árboles aleatorios para la selección de características, cambiando el número de componentes seleccionadas. El número de árboles utilizado para este experimento fue de 100.

Las 10 características más representativas de los datos son:

NN, IN, NNP, JJ, DT, VBZ, the, ., downcase, s

en orden decreciente de importancia, usando estas features se obtiene una precisión de 0.76 como se puede ver en la tabla 3.

Las 50 características más representativas de los datos son:

NN, IN, NNP, JJ, DT, VBZ, the, ., downcase, s, upcase, punct, NNS, VBD, VB, mixedcase, VBN, Wh, WRB, WP, Wha, hat, PRP, he, in, CD, th, t, a, f, VBP, of, is, 's, RB, TO, was, JJS, e, o, is, to, d, at, ow, CC, How, PRP$, wa, Who

Analizando las primeras 50 componentes podemos ver que tenemos los POS tag más usados, las expresiones regulares que me indican si está en mayúsculas, minúsculas o es signo de puntuación y también aparecen palabras como 'the', 'of', 'in', 'at', 'to', 'is', 'was', 'Who', 'How'.

4. Conclusiones

Se pudo diseñar un POS tagger especializado en preguntas, el cual cumple con todos los objetivos planteados.

La precisión obtenida supera a todos los POS taggers de la suite NLTK, aunque en la tabla 1 solo se muestra el POS tagger basado en HMM que fue el de mayor precisión de la suite NLTK y también se supero la precisión del tagger SENNA el cual esta basado en Redes Neuronales. Con respecto a la comparación con otros POS taggers de uso común faltaría probar el POS tagger de la librería freeling.

Una de las cuestiones que quedan abiertas es la posibilidad de manejar características faltantes (conocido como missing features en la literatura), para esto se debería pensar en otro modelo para las probabilidades de emisión, posiblemente basado en ensemble classifiers.

5. Mejoras a Futuro

Creemos que se pueden obtener mejores resultados con el agregado de nuevas características. Sería interesante el agregado de palabras funcionales como pueden ser preposiciones y verbos auxiliares, aunque con el esquema actual están contemplados las de tamaño 2 y 3 que aparecen usualmente. También se podrían agregar clases de palabras como pueden ser días de la semana, meses, etc. o también se puede plantear obtener estas clases de palabras usando algún algoritmo de clustering.

Referencias

[1] Y. Altun, I. Tsochantaridis, T. Hofmann, et al. Hidden markov support vector machines. In *ICML*, volume 3, pages 3–10, 2003.

[2] S. Bird, E. Klein, and E. Loper. Nltk, natural language toolkit. `http://nltk.org/`.

[3] S. Bird, E. Klein, and E. Loper. *Natural language processing with Python*. O'reilly, 2009.

[4] C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[5] R. Collobert. Senna natural language processing tool. `http://ronan.collobert.com/senna/`, 2011.

[6] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537, 2011.

[7] C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[8] H. Deng and G. Runger. Feature selection via regularized trees. In *Neural Networks (IJCNN), The 2012 International Joint Conference on*, pages 1–8. IEEE, 2012.

[9] J. Judge, A. Cahill, and J. Van Genabith. Questionbank: Creating a corpus of parse-annotated questions. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 497–504. Association for Computational Linguistics, 2006.

[10] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[11] J. Platt et al. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10(3):61–74, 1999.

[12] L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.