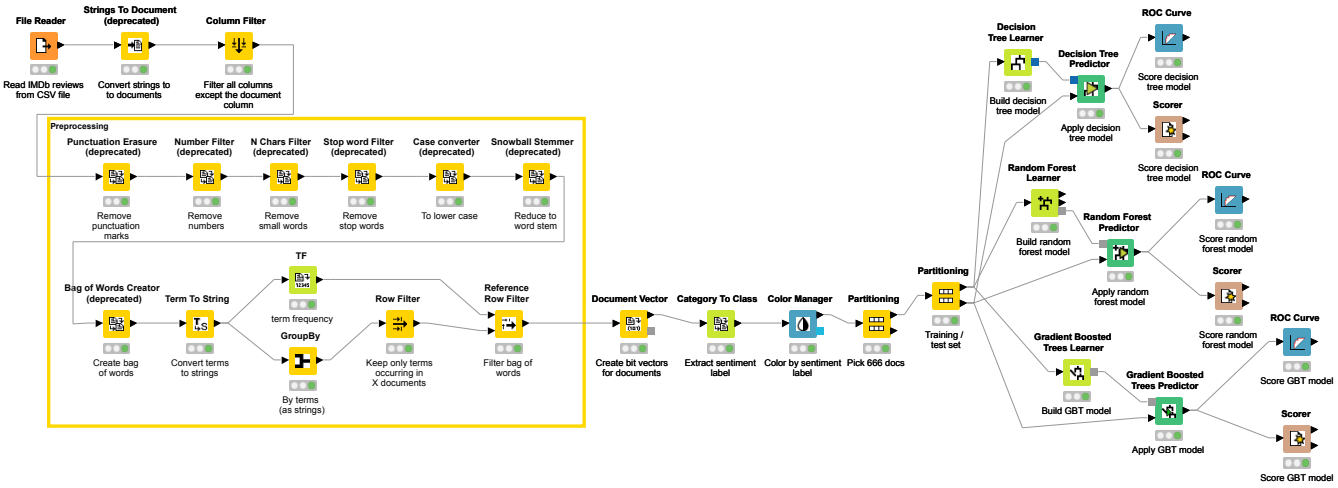
Práctica 2 Minería de Texto - Minería de Medios Sociales.

Curso: 2018/2019. Néstor Rodríguez Vico. DNI: 75573052C - nrv23@correo.ugr.es

20 de mayo de 2019

1. Clasificación.

Primero he resuelto la parte de clasificación. Para ello, he cogido el *workflow Sentiment Classification* proporcionado por el profesor de la asignatura y lo he modificado para añadir dos nuevos clasificadores. El *workflow* resultante es el siguiente:



Se han añadido los siguientes nodos:

- *Random Forest Learner*: Dicho nodo implementa un algoritmo *Random Forest*.
- *Random Forest Predictor*: Una ves hemos aprendido un modelo *Random Forest*, usamos este nodo para realizar la predicción.
- *Scorer (1)*: Evaluamos como ha ido la predicción del modelo *Random Forest*.
- *Gradient Boosted Trees Learner*: Dicho nodo implementa un algoritmo *Gradient Boosted Trees*.
- *Gradient Boosted Trees Predictor*: Una ves hemos aprendido un modelo *Gradient Boosted Trees*, usamos este nodo para realizar la predicción.
- *Scorer (2)*: Evaluamos como ha ido la predicción del modelo *Gradient Boosted Treest*.

El preprocesamiento seguido es el mismo que había en el *workflow* proporcionado por el profesor. Los pasos son:

- Eliminar signos de puntuación.
- Eliminar números.

- Eliminar palabras pequeñas.
- Eliminar *stopwords*.
- Convertir a minúsculas.
- Quedarnos con la raiz de las palabras.
- Construimos la bolsa de palabar y convertimos los términos a *strings*.
- Calculamos la frecuencia de términos (tf) y nos quedamos sólo con los términos que aparecen en, al menos, 20 documentos.
- Finalmente, filtramos la bolsa de palabras.

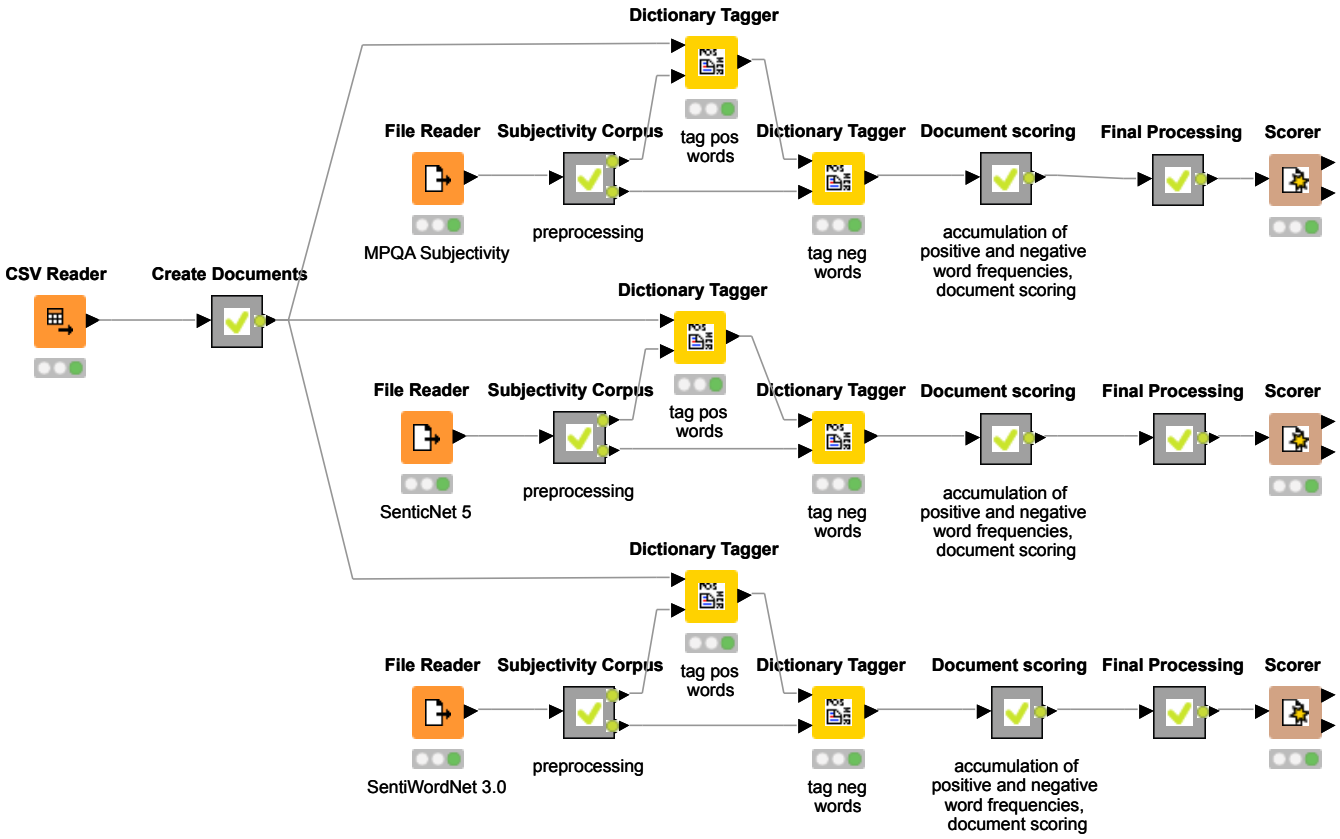
Como podemos ver en medio del *workflow*, hay dos nodos *partitioning*. El segundo de ellos es para separar el conjunto de datos en un conjunto de *train* (*70 %* de las instancias) y *test* (*30 %* de las intancias). El primero de ellos lo que hace ese seleciconar *666* documentos de forma estratíficada. Esto se ha hecho para tener *200* instancias en el conjunto de *test*, ya que es el número máximo de instancias que he podido ejecutar en la segunda parte de la práctica (comentaré los detalles en dicha sección). Los resultados obtenidos son los siguientes:

Modelo	**% Acierto**	**Kappa**
Decision Tree	83.5	0.67
Random Forest	85.0	0.70
Gradient Boosted Trees	89.0	0.78

Como podemos ver, el mejor resultado lo obtenemos con el algoritmo *Gradient Boosted Trees*.

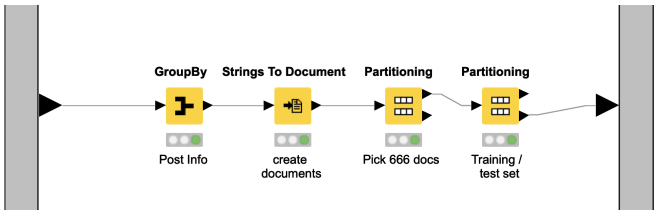
2. Análisis de sentimientos.

Para resolver esta parte he cogido el *workflow Sentiment Analysis* proporcionado por el profesor de la asignatura y lo he modificado adaptarlo al enunciado. El *workflow* resultante es el siguiente:



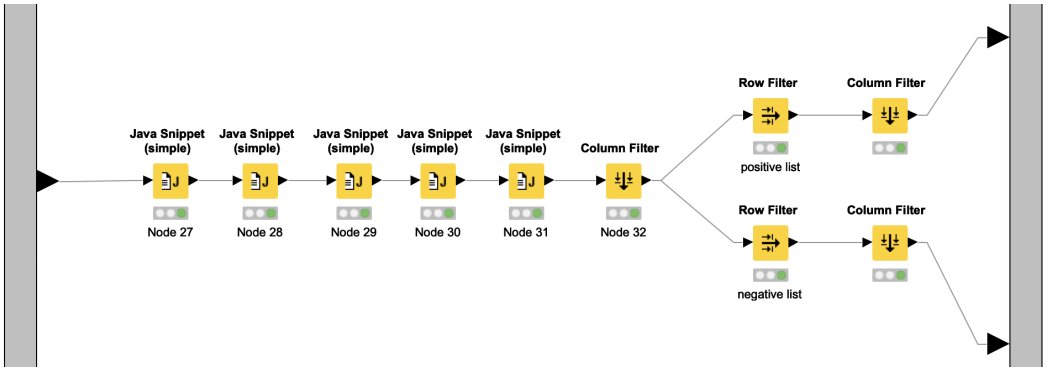
Las diferentes partes del *workflow* son:

- En la primera parte podemos ver la parte común a las tres ramas posteriores. Dicha parte se encarga de leer el conjunto de datos, y ejecutar el nodo *Create Documents*. Dicho nodo es el siguiente:



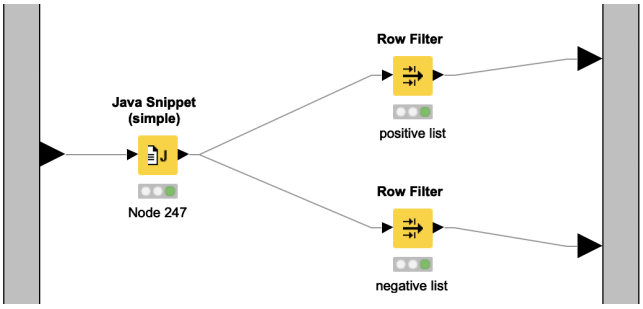
Dicho nodo agrupa por identificador (por si hay varios documentos con el mismo identificador) y convierte el texto leido a una variable de tipo *Document*, que es la usada por *KNIME* para representar un documento. A continuación, cogemos los mismos *666* documentos que hemos cogido en la práctica de Clasificación y dividimos el conjunto de dichos documentos en train y test. Tanto en la parte de clasificación como en la parte de análisis de sentimientos se han usado las mismas semillas, lo cual nos garantiza que el conjunto de documentos usado como conjunto de test en la parte de clasificación es el mismo conjunto de instancias que estamos usando en la parte de análisis, pudiendo hacer así una comparación justa. Como comenté anteriormente, para este *workflow* he usado *200* documentos sólo. Esto se debe a que, por problemas de rendimiento de mi máquina, no he podido ejecutar el *workflow* para más elementos.

  - La primera rama hace uso del *lexicon MPQA Subjectivity*. Tras leer el *lexicon*, he aplicado preprocesamiento. Dicho preprocesamiento es el siguiente:



Este preprocesamiento consiste en utilizar nodos del tipo *Java Snippet* para parsear las columnas del *corpus* eliminando los símbolos *igual* (=) de cada columna para quedarnos sólo con la información relevante. A continuación, separamos el conjunto de documentos en documentos con sentimiento positivo (salida superior del nodo) y documentos con sentimiento negativo (salida inferior del nodo).

- La segunda rama hace uso del *lexicon SenticNet 5*. Tras leer el *lexicon*,, he aplicado preprocesamiento. Dicho preprocesamiento consiste en separar el conjunto de documentos en documentos con sentimiento positivo (salida superior del nodo) y documentos con sentimiento negativo (salida inferior del nodo).
- La tercera rama hace uso del *lexicon SentiWordNet 3.0*. Para poder leer correctamente dicho *lexicon*, he hecho un script en *Python* [1] para formatear un poco el *lexicon*. El problema que tienes dicho lexicon es en la columna *SynsetTerms*. Dicha columna contiene el término en cuestión, el problema es que puede contener una lista de términos, los cuales son sinónimos entre ellos. Por lo tanto, lo que he hecho es repetir cada fila tantas veces como términos haya dejando en cada fila un único término. Tras leer el *lexicon*, he aplicado preprocesamiento. Dicho preprocesamiento es el siguiente:

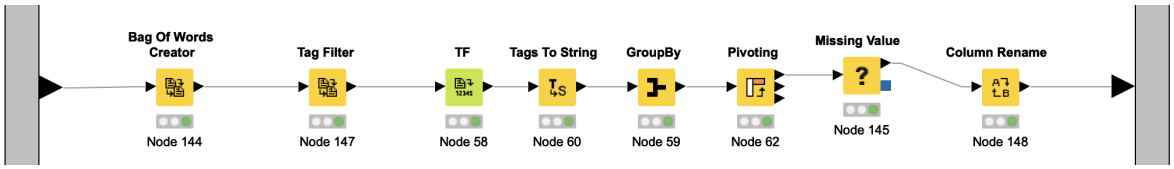


Lo primero que hacemos es usar un nodo *Java Snippet*. Dicho nodo lo que hace es calcular la polaridad de las palabras del *lexicon*, ya que este no nos las da, sino que nos da una intensidad de positividad y otra de negatividad. Lo que he hecho es considerar una polaridad positiva si la intensidad de positividad es mayor que la intensidad de negatividad y una polaridad negativa en caso contrario. A continuación, separo el conjunto de documentos en documentos con sentimiento positivo (salida superior del nodo) y documentos con sentimiento negativo (salida inferior del nodo).

- Una vez tenemos la estructura del *corpus* lista, realizamos los mismos pasos para todos ellos. Los dos primeros nodos son nodos de etiquetado (*Dictionary Tagger*), los cuales sirven para etiquetar las elementos de los documentos en base al *corpus* usado.

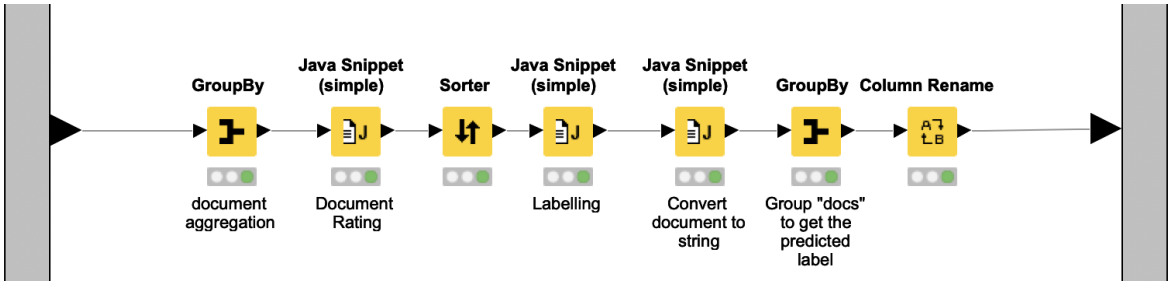
[1] Dicho *script* no se ha entregado. Si lo desea, hágamelo saber y se lo mando.

- El siguiente paso es puntuar el documento. Para ello hacemos uso del *metanodo Document Scoring*. Dicho nodo es similar al presente en el *workflow* original. En dicho nodo hacemos lo siguiente:



  Primero se construye una bolsa de palabras y filtrmos los términos no deseados, calculamos la frecuencia de términos (tf), convertimos las etiquetas a *strings*, agrupamos la salida por términos e imputamos los valores de frecuencia de términos perdidos si los hubiese. Finalmente, renombramos las columnas para que sea más fácil hacer referencias a ellas posteriormente.

- A continuación, llamamos al metanodo *Final Processing*. Dicho nodo hace lo siguiente:



  En el nodo *GroupBy* agrupamos por documentos. En el nodo *Java Snippet* sumamos la puntuación de los término dentro de cada documentos y ordenamos el resultado con el nodo *Sorter*. En el siguiente nodo *Java Snippet* lo que hacemos es cambiar el valor numérico de la puntuación por una etiqueta. Si dicha puntuación es mayor que cero la etiqueta será *POS*, *NEG* en caso contrario. El segundo nodo *Java Snippet* convierte la variable que almacena el documento a una variable de tipo *String*, para poder manipularla correctamente en etapas futuras. Finalmente, agrupamos por documento para obtener la etiqeuta predicha de cada uno de los documentos y renombramos las columnas para que sea más fácil identificar que representa cada una de ellas.

- Una vez tenemos la etiqueta predicha y la etiqueta real, usamos el nodo *Scorer* para compararlas y sacar las métricas que se muestran a continuación.

Los resultados obtenidos son los siguientes:

Lexicon	% Acierto	Kappa
MPQA Subjectivity	70.5	0.399
SenticNet 5	53.0	0.011
SentiWordNet 3.0	62.7	0.018

Como podemos ver, los mejores resultados los obtenemos con el lexicon *MPQA Subjectivity*, el cual obtiene unos resultados bastante mejores que los obtenidos con los otros dos lexicones.

3. Comparación.

Tal y como he comentado en la sección de análisis, cuando se han seleccionado los documentos, se ha garantizado que el conjunto de *test* en la parte de clasificación contiene los mismos documentos que el

conjunto usado en la parte de análisis. A continuación podemos ver una tabla comparativa de los resultados obtenidos aplicando las distintas técnicas

Modelo	**% Acierto**	**Kappa**
Decision Tree	83.5	0.67
Random Forest	85.0	0.70
Gradient Boosted Trees	89.0	0.78
Media	85.866	0.716

Lexicon	**% Acierto**	**Kappa**
MPQA Subjectivity	70.5	0.399
SenticNet 5	53.0	0.011
SentiWordNet 3.0	62.7	0.018
Media	62.066	0.142

Cómo podemos ver, las técnicas de clasificación son bastante mejores, obteniendo un porcentaje de acierto del *85.866 %* en media frente al *62.066 %* en media obtenido por las técnicas de análisis.