# Instalação e Configuração do Eclipse para desenvolvimento de nodes do tipo textprocessing no KNIME

Daniel Andrade Costa Silva (daniacs@gmail.com)
Orientação e ajuda: Cristiano Carvalho (cristiano.dcc@gmail.com)

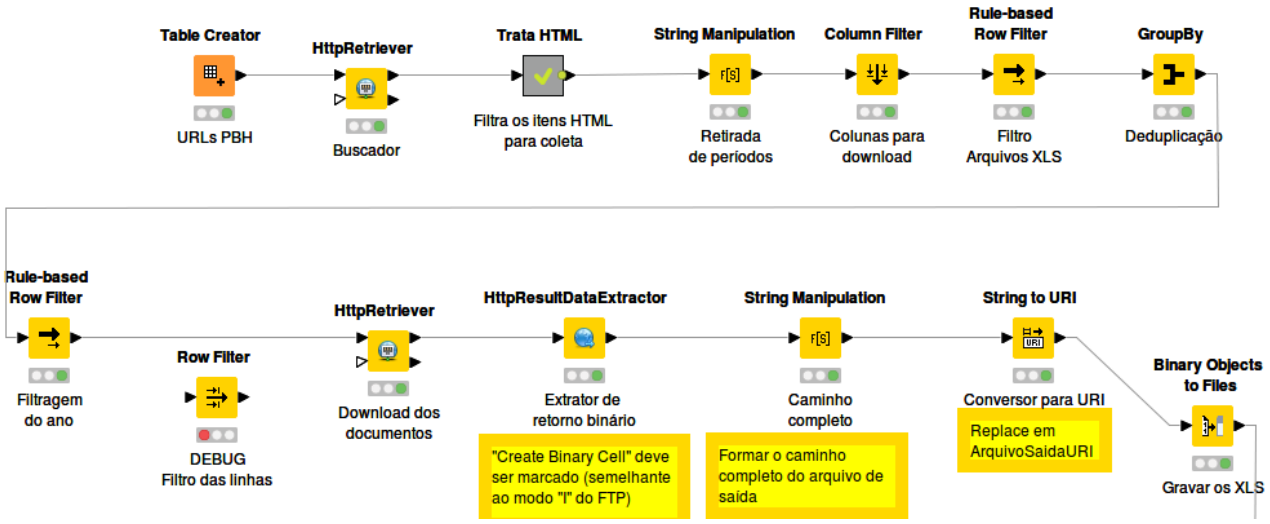
Belo Horizonte, Minas Gerais, Brasil – Ago/2017

# Sumário

# 1 Objetivo e Conceitos

KNIME (Konstanz Information Miner - https://www.knime.com/) é uma ferramenta de código aberto (open source) que permite ao usuário executar tarefas de Mineração de Dados, Mineração de Textos e análises diversas, através de uma interface bastante intuitiva.

O usuário constrói um workflow (fluxograma) onde cada node (nó) desempenha uma tarefa, que pode ser buscar dados de um site, transformar caracteres de caixa alta para caixa baixa, retirar pontuação de textos, substituir strings, fazer análises estatísticas de tabelas, entre outras várias implementadas pela ferramenta.




O objetivo desta documentação é descrever, de forma **bem** concisa, o ambiente de desenvolvimento, construção e testes de um node do tipo POS Tagger para o idioma português utilizando as bibliotecas do Apache OpenNLP. Parte-se do pressuposto que o leitor já tenha usado o KNIME, tenha noções de programação na linguagem Java e alguns conceitos básicos do Eclipse.

POS tagging, Part-of-speech tagging, rotulagem ou etiquetagem ou classificação morfossintática é o processo de classificar um termo de uma frase em sua respectiva classe gramatical. Por exemplo, na frase “O vento arrastou casas durante a tempestade”, seria algo do tipo:

O (artigo) vento (substantivo) arrastou (verbo) casas (substantivo) durante (preposição) a (artigo) tempestade (substantivo) .(pontuação)

O processo pode ser utilizado de várias formas interessantes: classificação de textos conforme frequência de seus termos, automatização de atendimentos, corretores ortográficos e várias outras.

Ferramentas como o KNIME permitem a execução desse processo de forma que o usuário não precise saber codificar um programa. No entanto, até o momento ainda não há um node do KNIME que faça essa tarefa no idioma português, apenas inglês e alemão.

É importante ressaltar que este trabalho não é um POS Tagger nativo da ferramenta. O objetivo é permitir que o leitor saiba como configurar o ambiente de desenvolvimento de forma que ele possa construir outros nodes conforme suas necessidades, sejam elas acadêmicas, comerciais, pessoais, etc.

# 2 Eclipse

## *2.1 Instalação e atualização*

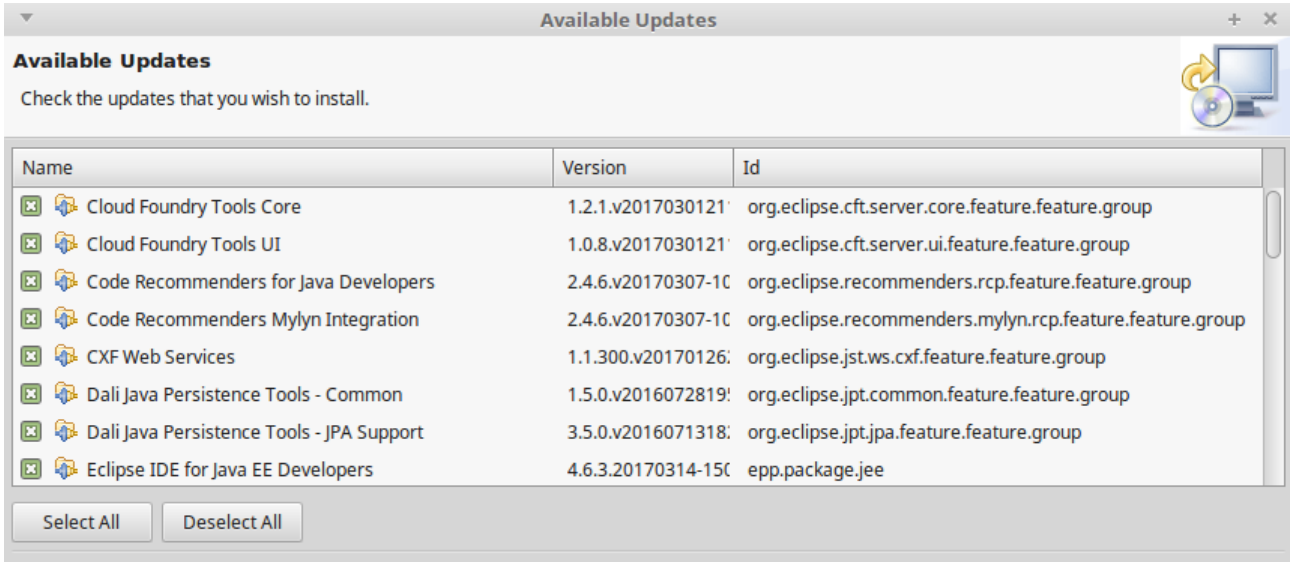
- Instalar e executar o Eclipse **Neon**. No momento desta documentação, o Oxygen ainda estava na Release Candidate 3. Pode ser que para o Oxygen o processo seja o mesmo, mas ainda não foi testado.
  - Download em http://www.eclipse.org/downloads/packages/release/Neon/3
  - Eclipse IDE for Java EE Developers

- Descompactar e executar o Eclipse

```
$ tar xvzf eclipse-jee-neon-3-linux-gtk-x86_64.tar.gz
$ cd eclipse
$ ./eclipse
```

- Definir o diretório onde estará o espaço de trabalho (Workspace)
  - /home/daniel/Documents/knime-devel/workspace

- Atualizar o eclipse. Caso houver atualizações, instalar todas.
  - Selecionar a opção **Help → Check for Updates**

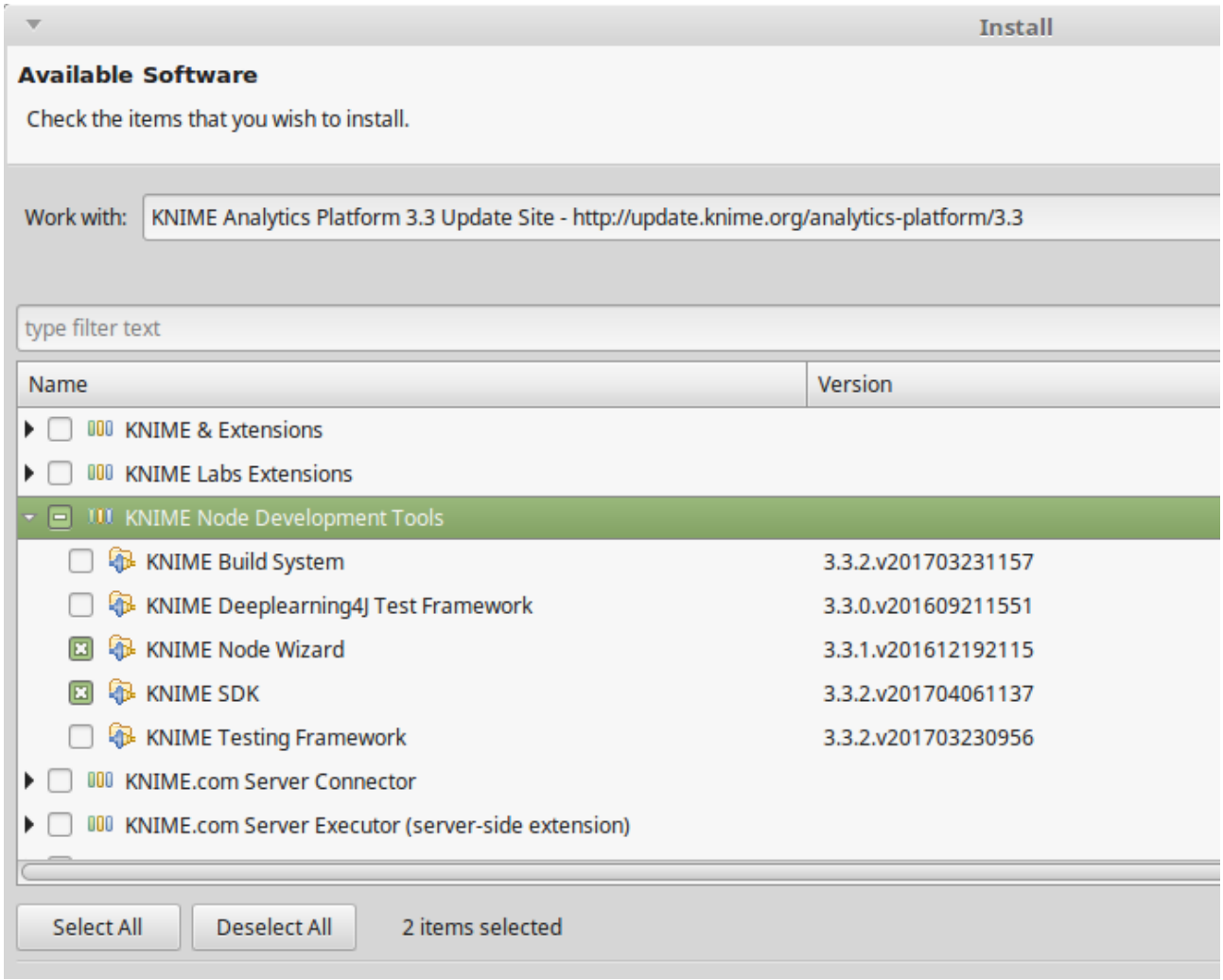



  - Next → Next → Aceitar os acordos da Licença → Finish

## 2.2 Configuração para desenvolvimento de nodes do KNIME

- Adicionar o repositório para os itens do KNIME
  - Selecionar a opção **Help → Install New Software → Add**
  - Vai aparecer uma caixa de diálogo. Preencher os seguintes campos:
    - **Name**: KNIME Analytics Platform 3.3 Update Site
    - **Location**: http://update.knime.org/analytics-platform/3.3

- Instalar os seguintes plugins para desenvolvimento de nodes:
  - KNIME Node Development Tools
    - KNIME Node Wizard
    - KNIME SDK




"KNIME Node Wizard" provê facilidades como a criação de modelos de classes durante a criação de projetos de novos nodes e "KNIME SDK" instala uma versão mínima da aplicação para testes de execução no Eclipse.

## 2.3 Os plugins de textprocessing

São necessários para manipular dados do tipo Documento. Os plugins instalados no Eclipse para permitir esse tipo de manipulação foram os seguintes:

- KNIME Textprocessing (org.knime.features.ext.textprocessing.feature.group): necessário para que o Eclipse ache o caminho dos “imports”.

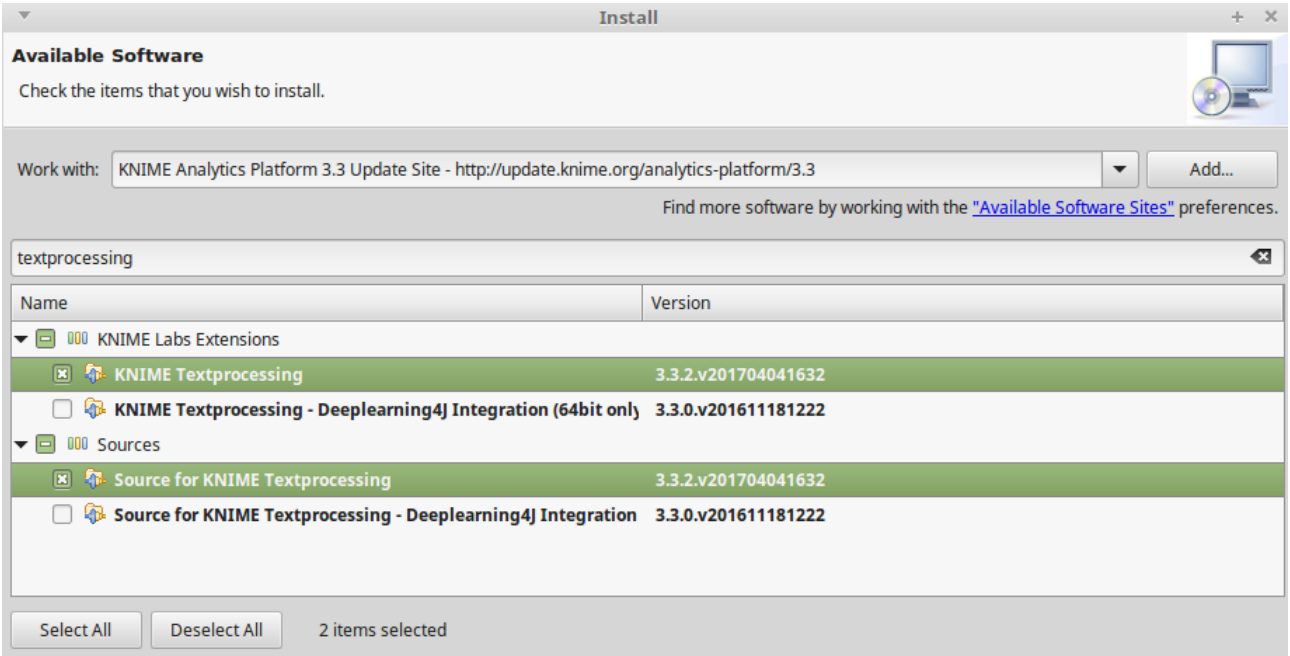
→ Referência de biblioteca (Projeto → Build Path → Configure Build Path → Add External JARs →/home/daniel/Programas/eclipse/plugins/org.knime.ext.textprocessing_3.3.2.v201704041632/knime-textprocessing.jar)

No entanto, não instala o Javadoc necessário.

- Source for KNIME Textprocessing (org.knime.features.ext.textprocessing.source.feature.group): no código fonte, basta segurar Ctrl + clique em uma classe qualquer do textprocessing. Na tela de erro, Attach Source → External Location → External File → /home/daniel/Programas/eclipse/plugins/org.knime.ext.textprocessing.source_3.3.2.v201704041632.jar

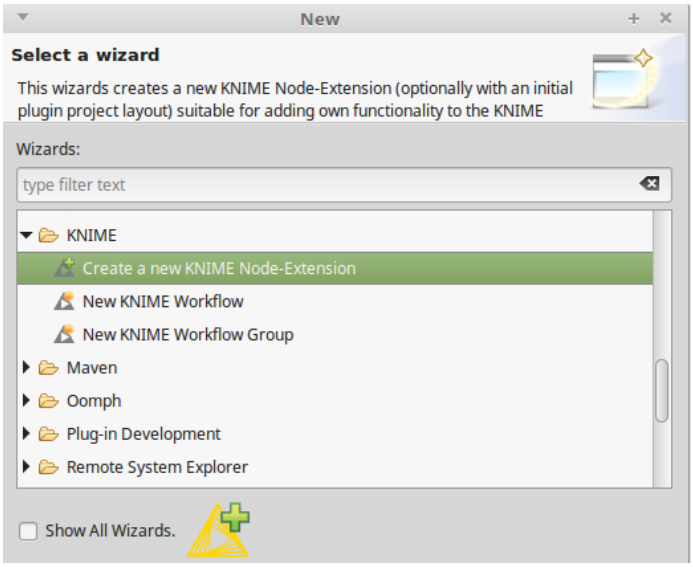
Opção help → Install New Software → Work With: KNIME Analytics Platform

Os plugins podem ser encontrados facilmente usando o filtro “textprocessing”, conforme figura abaixo:

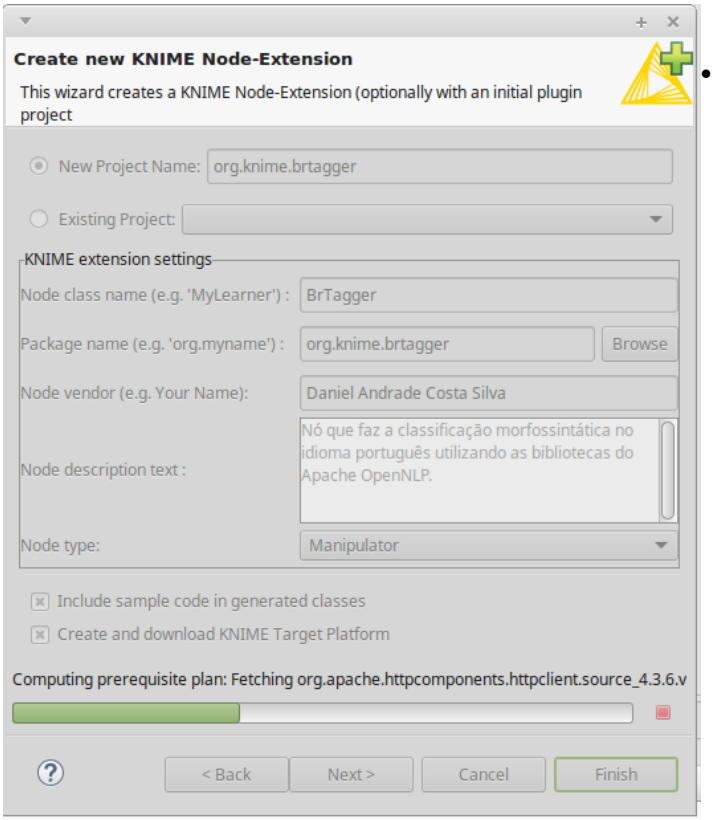


# 3 Criação do projeto do novo node

- Selecionar File → New → Other → KNIME → Create a new KNIME Node-Extension




- As opções devem ser preenchidas de acordo com o objetivo do node. Em geral, os nodes são do tipo “Manipulator”, o que significa que farão a manipulação de dados.



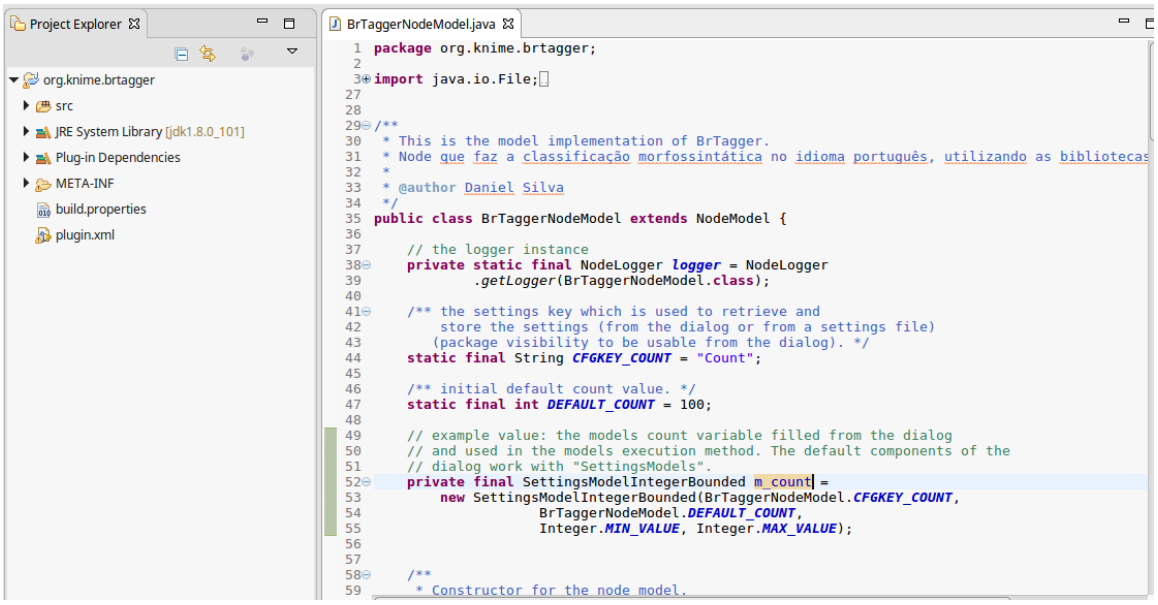

- Na criação do projeto, selecionar as duas opções mais abaixo (“Include sample code in generated classes” e “Create and download KNIME Target Platform”).

Exemplo:

New Project Name: org.knime.brtagger
Node Class name: BrTagger
Package name: org.knime.brtagger
Node vendor: <Nome do desenvolvedor>
Description: Node que faz a classificação morfossintática no idioma português, utilizando as bibliotecas do Apache OpenNLP.
Node Type: Manipulator

O tamanho do download do KNIME Target Platform é ~350MB.
O download é feito no workspace do usuário e não no diretório de instalação do Eclipse.
Os arquivos baixados ficam em /.metadata/.plugins/org.eclipse.pde.core/.bundle_pool/plugins.

Após o download da plataforma e estruturação do projeto, o Eclipse exibe a tela inicial do projeto já com a classe BrTaggerNodeModel para ser codificada.




Os nodes são compostos por quatro classes principais:

- **NodeModel**: responsável por executar o algoritmo principal do node, que implementa sua lógica. Toda manipulação de dados é feita nesta classe. Além disso, administra o fluxo de dados (entrada e saída) do node, definindo as estruturas de dados que serão processadas e quais serão retornadas ao próximo node.

- **NodeDialog**: implementa uma caixa de diálogo (dialog) utilizada para definir os parâmetros de execução do node, caso haja. Sempre que o usuário configura um node, o KNIME carrega a classe NodeDialog correspondente, permitindo, por exemplo, que o usuário possa escolher entre dois algoritmos diferentes para executar a mesma tarefa. É opcional, visto que um node pode executar uma tarefa simples, sem necessidade de configurações específicas.

- **NodeView**: exibe os dados da classe NodeModel de uma forma específica (além da tabela de resultados já exibida pelo próprio NodeModel). Pode ser utilizada para implementação de Cloud Tags, visualizadores de documentos, de imagens, de gráficos e quaisquer tipos de visualização dos dados consumidos ou produzidos pela NodeModel.

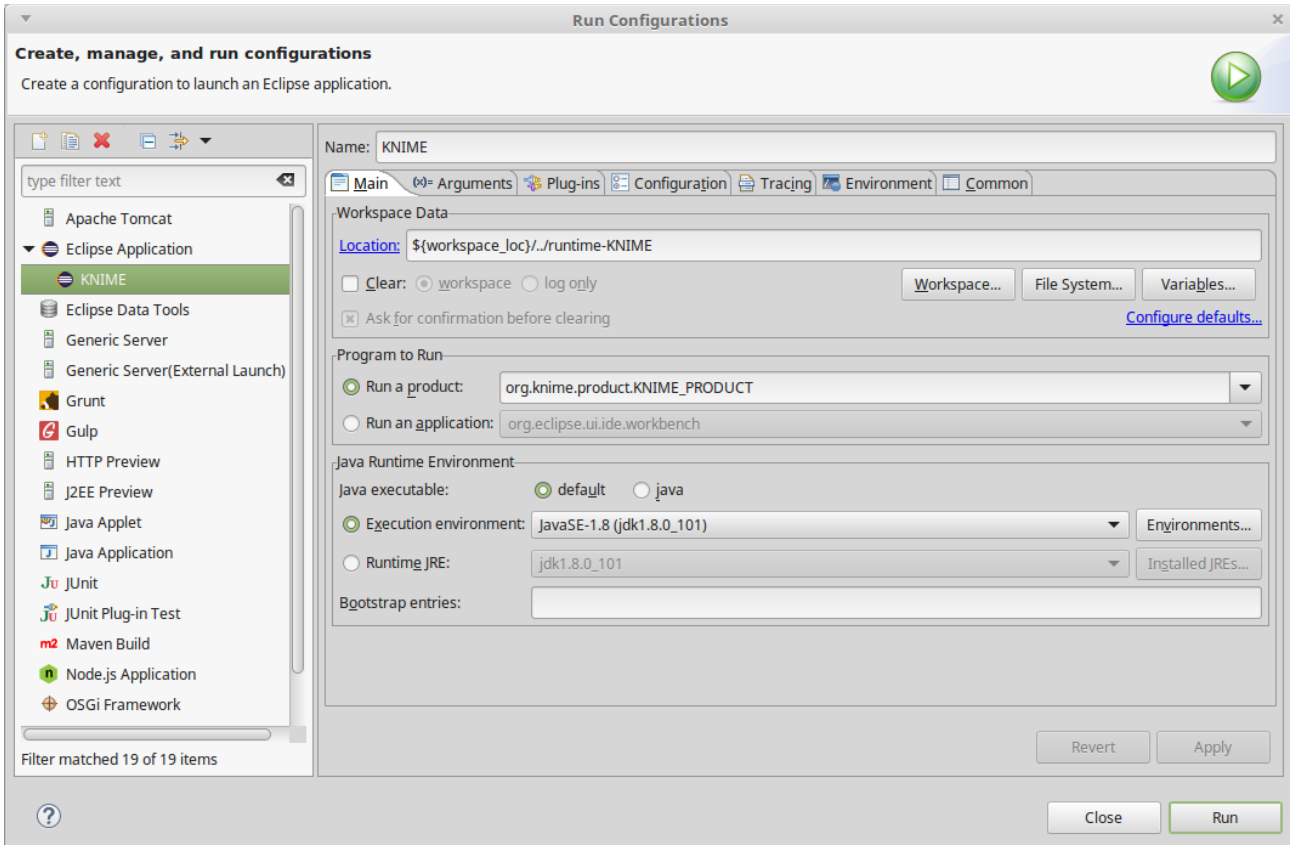
- **NodeFactory**: classe responsável por empacotar as classes NodeModel, NodeDialog e NodeView e registrar o node através do ponto de extensão org.knime.workbench.repository.nodes para que o node apareça no repositório de nodes.

# 4 Preparação do ambiente de execução (Target Platform)

## *4.1 Criação do ambiente de execução*

A compilação de um programa Java gera um objeto binário (.class) que pode ser executado através da JVM. No caso do plugin do KNIME, o objeto gerado é um conjunto de objetos .class, empacotado em um arquivo do tipo JAR e que deve ser executado no contexto da aplicação, ou seja, dentro do programa KNIME. Os seguintes passos explicam a configuração do ambiente de execução (ou *Target Platform*).

- Selecionar a opção Run → Run Configurations → Eclipse Application → New
  - Location: local onde será configurado o ambiente *runtime*;
  - Program to Run: Run a product → **org.knime.product.KNIME_PRODUCT**




- A partir desta configuração, já é possível executar o KNIME com o modelo de plugin gerado pelo Eclipse através da opção **Run → Run Configurations → selecionar a opção KNIME e clicar em Run**. O plugin BrTagger aparecerá como sendo o último, não categorizado. No entanto, o ambiente de execução é uma versão mínima do KNIME, que não contém os plugins de textprocessing, como “Strings to Document” ou o “POS Tagger”.

- Durante a configuração do ambiente de execução, na aba “Plug-ins”, o item “Launch with” possui a opção “all workspace and enabled target plug-ins” (que é selecionado por padrão). De acordo com a página de ajuda do Eclipse (http://help.eclipse.org/neon/index.jsp?topic=%2Forg.eclipse.pde.doc.user%2Fguide%2Ftools%2Flaunchers%2Fplugins.htm):

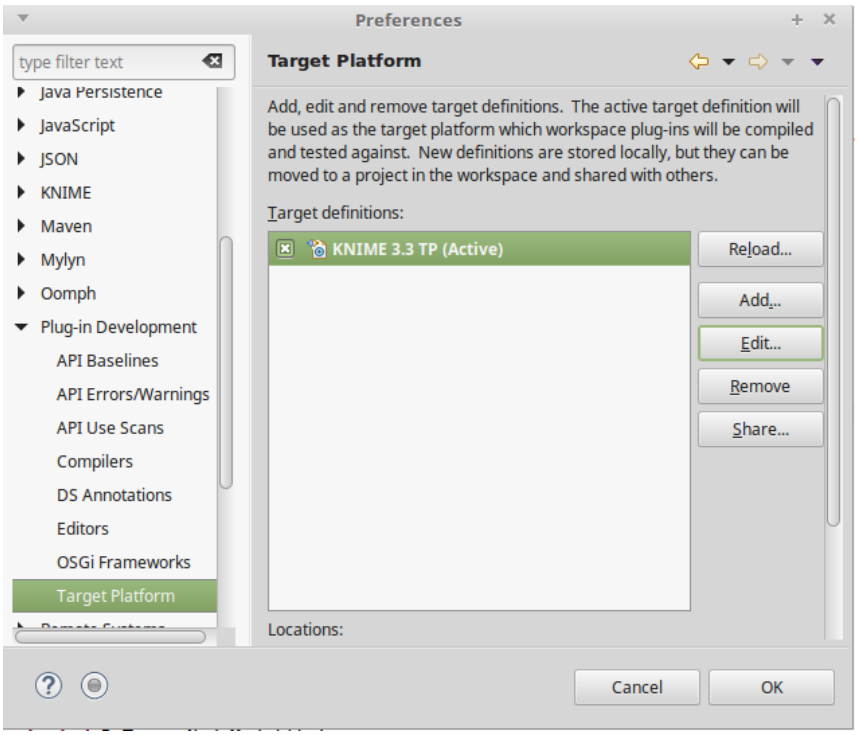
“The all workspace and enabled external plug-ins option is the default setting. With this option, the runtime Eclipse application you are launching will be made up of all the workspace plug-ins and all the plug-ins that are explicitly checked on the Window > Preferences... > Plug-in Development > Target Platform preference page. This option requires little maintenance because the list of plug-ins is maintained by PDE and updated prior to every launch as plug-ins are created or deleted in the workspace. This option is recommended when the Program to Run setting on the Main tab of the launch configuration is set to the default product or default application of your target.”

Ou seja, é necessário incluir os plugins do textprocessing e quaisquer outros plugins que devam ser carregadosconforme explicado a seguir.

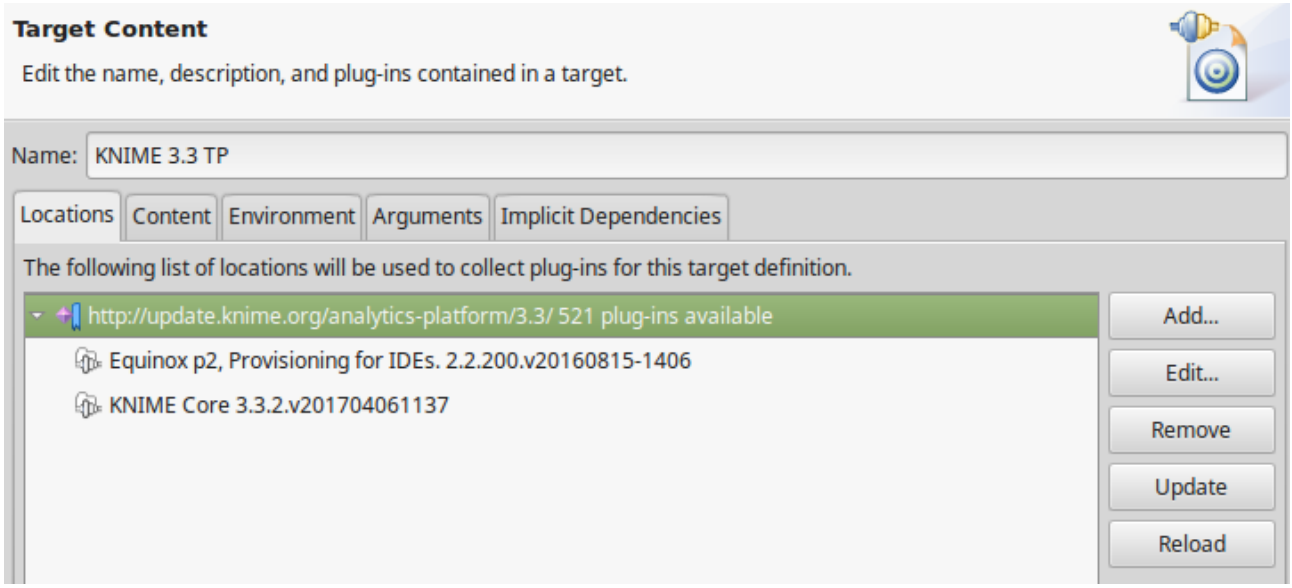
## 4.2 Inclusão do TextProcessing no ambiente de execução

Os plugins instalados nessa etapa podem ser carregados dentro da aplicação que o eclipse roda (no caso desta documentação, o KNIME).

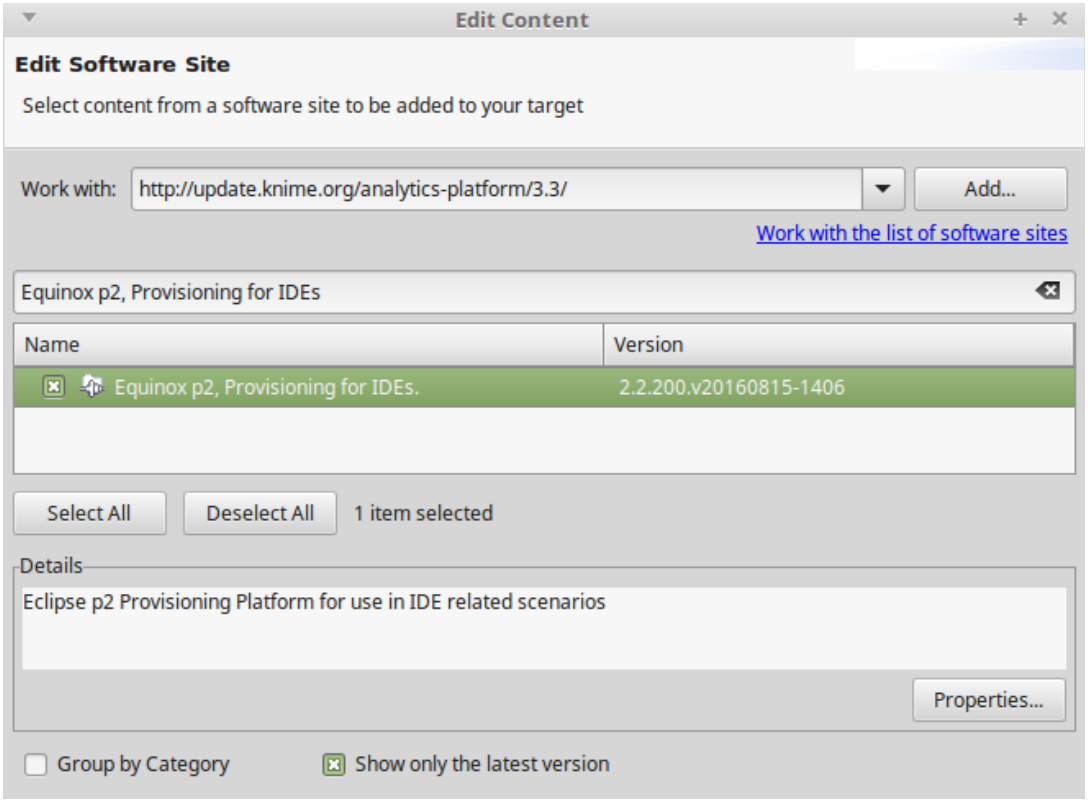
- Selecionar a opção: Window → Preferences → Plug-in Development → Target Platform → Selecionar a plataforma do KNIME→ Edit (e esperar o configurador carregar todas as opções).



- Selecionar o item http://update.knime.org/analytics-platform/3.3/ e clicar em “Edit”

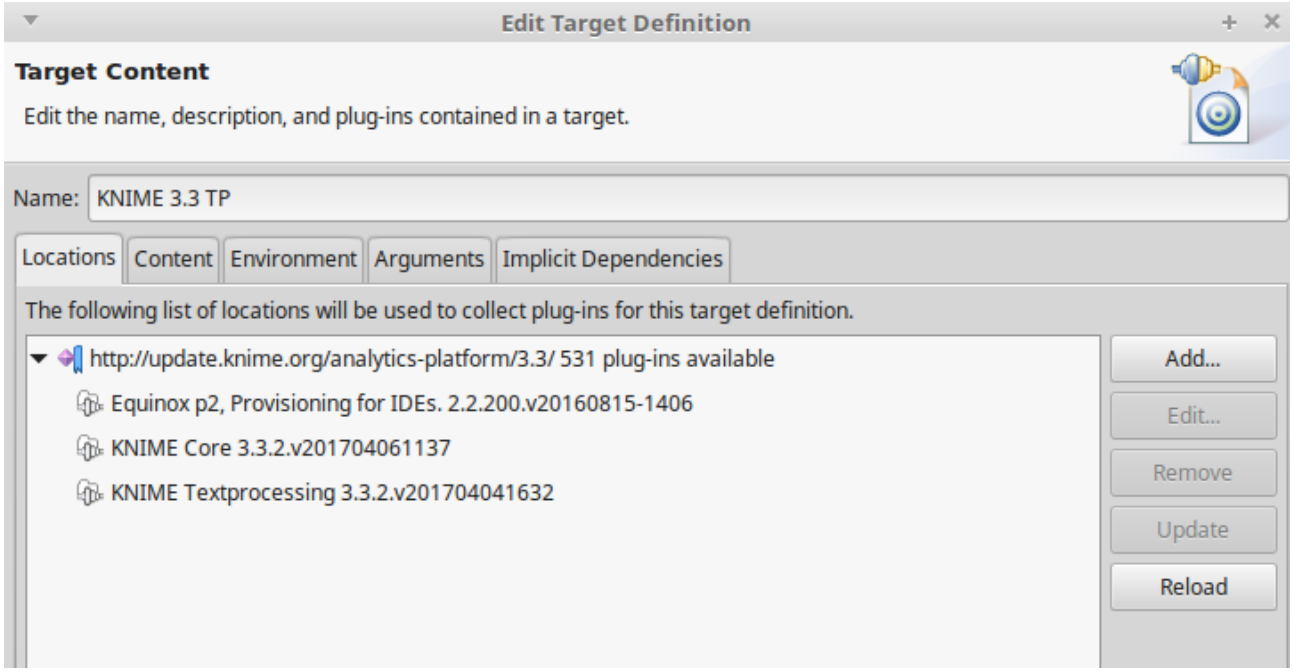



- Na tela “Edit Content”, desmarcar a opção **Group by Category** e marcar os plugins **Equinox p2, Provisioning for IDEs**, **KNIME Core** e **KNIME Textprocessing** (utilizar o filtro para encontrá-los mais facilmente).

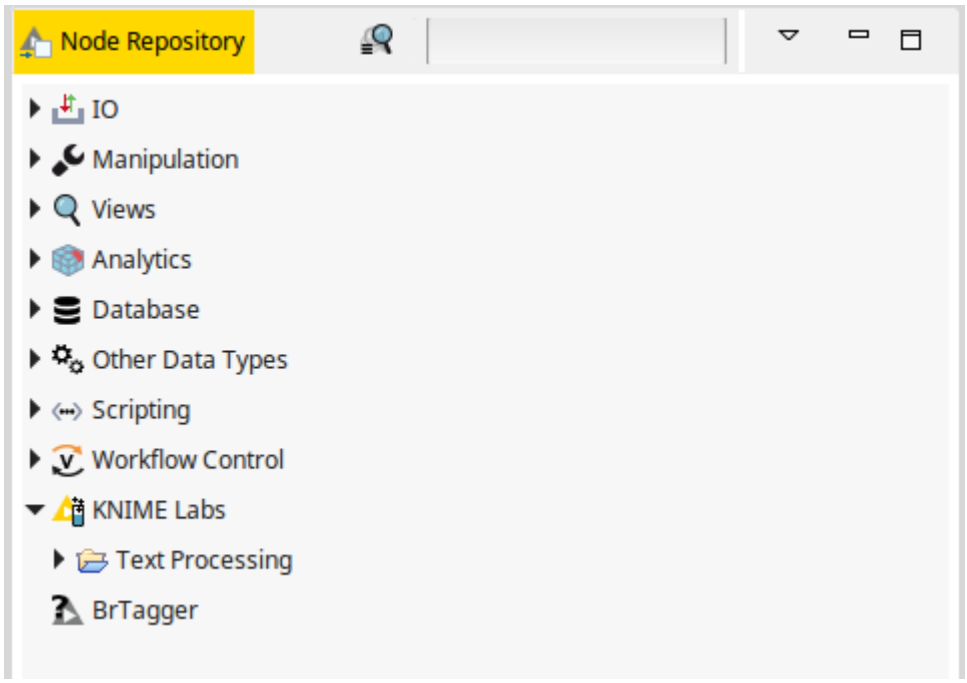



- Durante a seleção, deve aparecer “3 items selected”. Clicar em Finish pra instalar. Esperar instalar e clicar em Finish.

- A tela abaixo deve ser exibida. A partir daí, é possível carregar a extensão TextProcessing no KNIME para testes, em tempo de execução! ;)



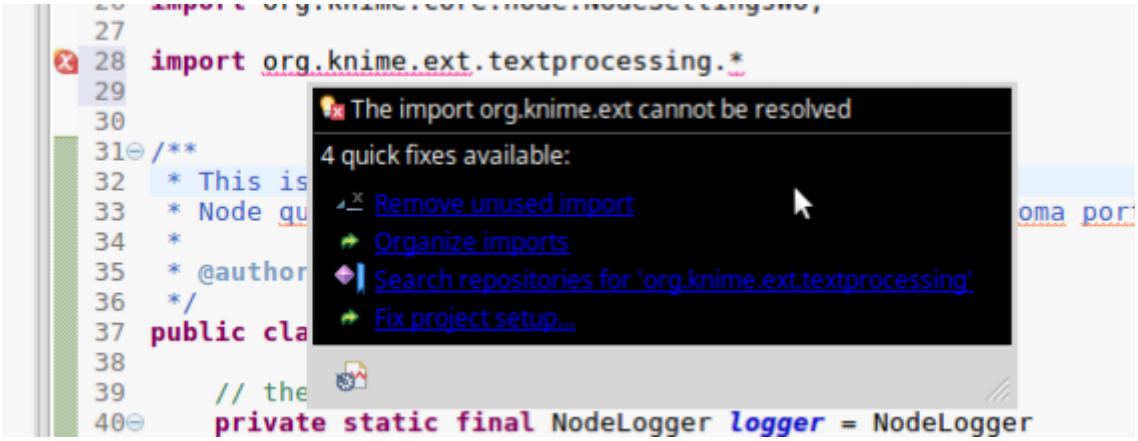

- Após o término do processo, pode-se usar o atalho Ctrl-F11 para executar o KNIME. No painél Node Repository, deve ser possível visualizar os nodes do Text Processing:



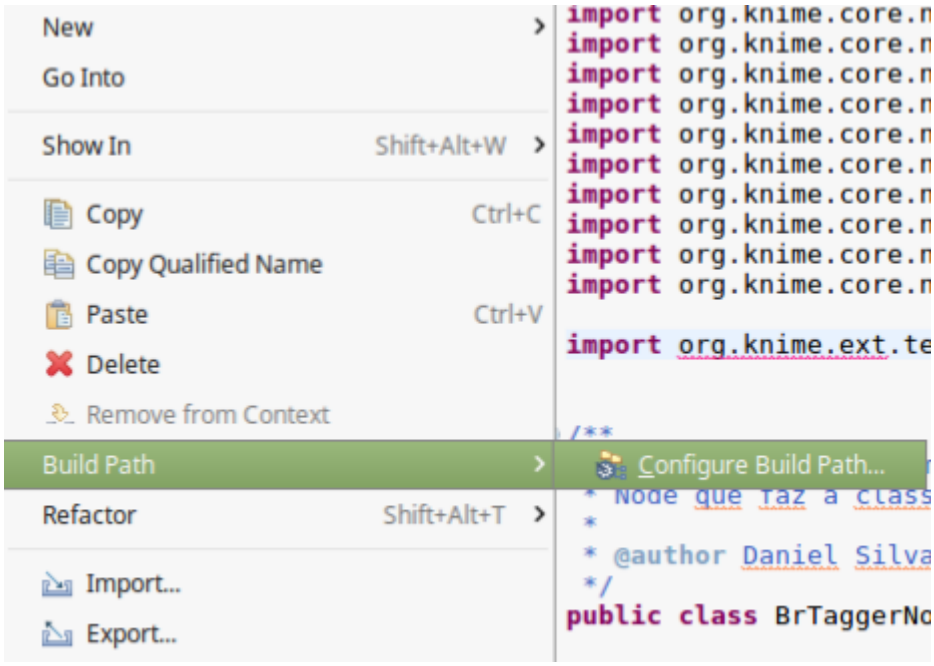
# 5 Codificação e construção

## *5.1 Incluir as referências ao TextProcessing no código*

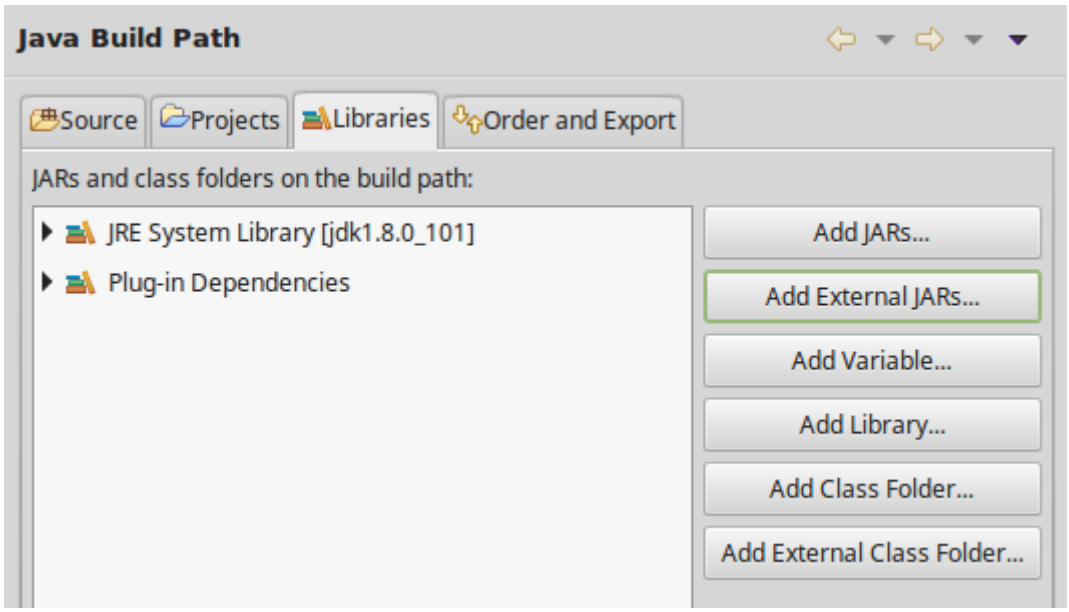
Ao tentar incluir as bibliotecas do textprocessing (necessárias para manipulação de documentos), o Eclipse não acha as referências.



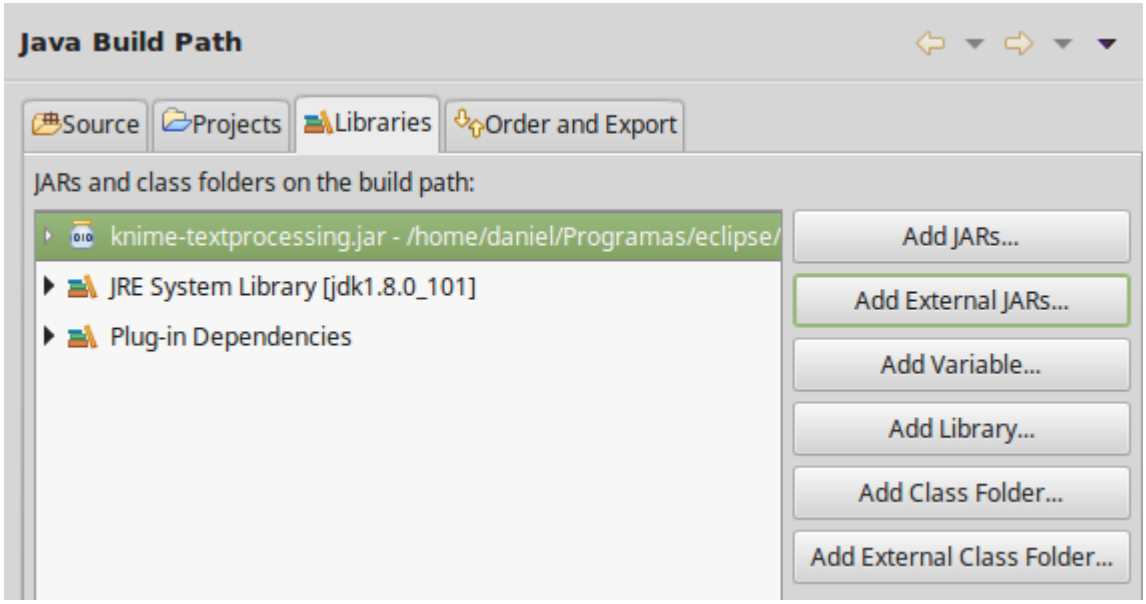

Clicar com o botão direito no projeto → Build Path → Configure Build Path...



- Escolher a aba Libraries e a opção Add External Jars




- Selecionar o plugin textprocessing da instalação do Eclipse: $ECLIPSE_HOME/plugins/org.knime.ext.textprocessing_<VERSAO>/knime-textprocessing.jar




- Clicar em OK. O erro no **import** irá desaparecer, indicando que a referência ao textprocessing foi bem sucedida.

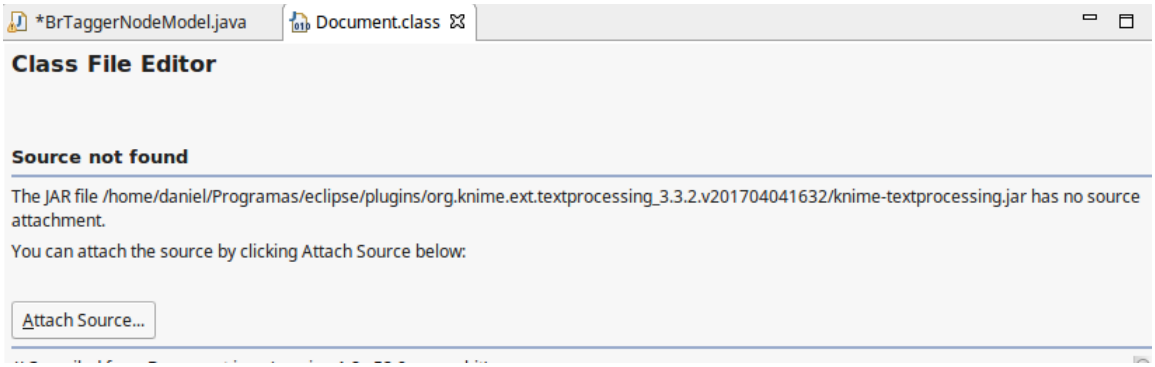
## 5.2 Incluir a documentação Javadoc do textprocessing

Em alguns casos é importante ter a documentação dos métodos utilizados pelo textprocessing. Para isso, basta utilizar qualquer classe do pacote para fazer a inclusão.

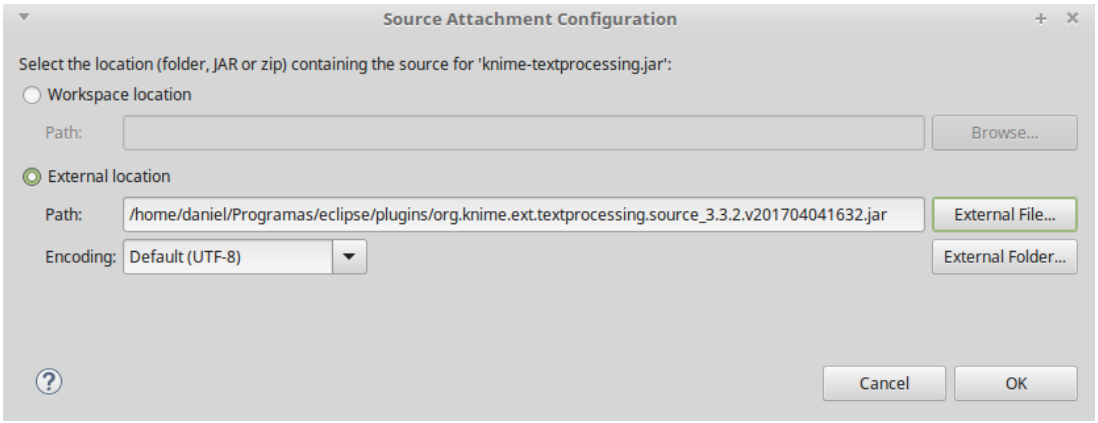
- Exemplo: no método execute, na classe `BrTaggerNodeModel`, declarar um objeto da classe Document. O Eclipse vai reclamar que a classe não existe e sugere importar a classe `import org.knime.ext.textprocessing.data.Document`. Seguir a sugestão. No código, segurar a tecla ctrl e clicar sobre a classe do textprocessing.

```
@Override
protected BufferedDataTable[] execute(final BufferedDataTable[] inData,
        final ExecutionContext exec) throws Exception {

    // TODO do something here
    logger.info("Node Model Stub... this is not yet implemented !");

    Document doc;
```

- O eclipse abre a tela dizendo que não foi possível encontrar a documentação. Anexar a documentação clicando em “Attach Source...”



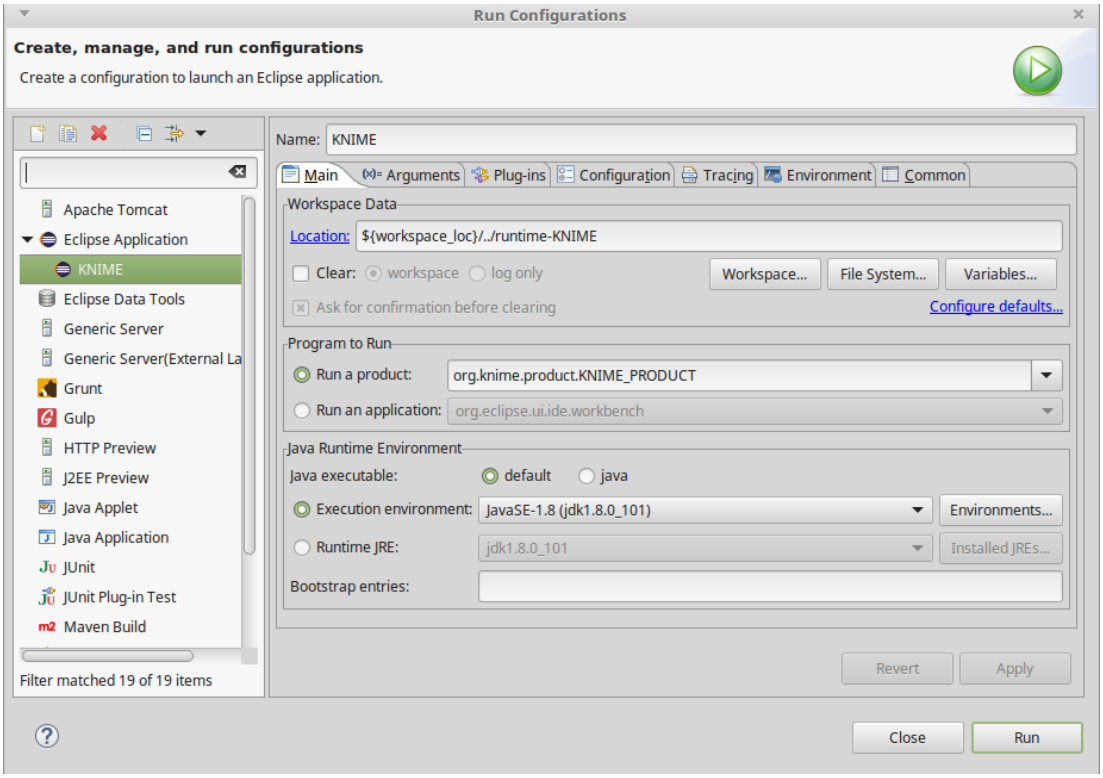

- Anexar o JAR do código fonte do textprocessing, no diretório de instalação do eclipse ($ECLIPSE_HOME/plugins/org.knime.ext.textprocessing.source_<VERSAO>.jar).



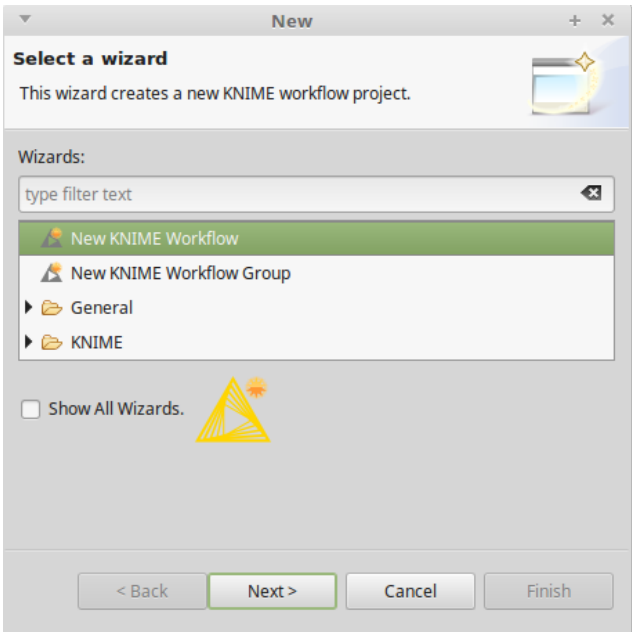

- Após clicar em OK, a documentação do código fonte do textprocessing estará disponível.

## 5.3 Leitura do documento

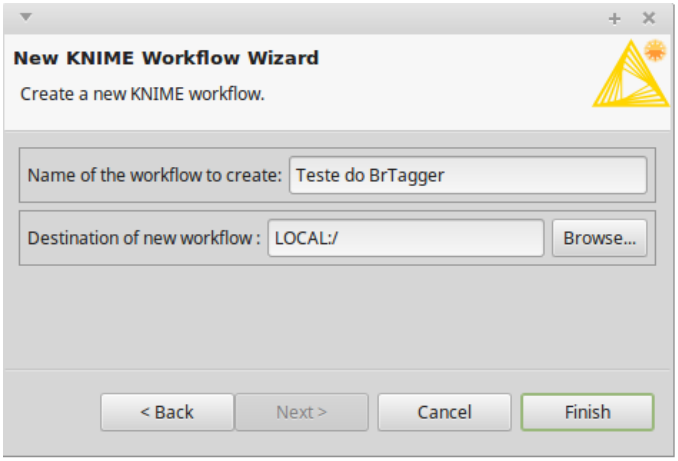
- Executar o KNIME a partir do Eclipse: Run → Run Configurations →KNIME → Run




- No KNIME, escolher o menu File → New → New KNIME Workflow



- Dar um nome ao Workflow e selecionar seu destino como sendo LOCAL:/




- Criar um workflow contendo 3 nodes: Table Creator, Strings to Document e o BrTagger (é o último node, visto que não foi categorizado).

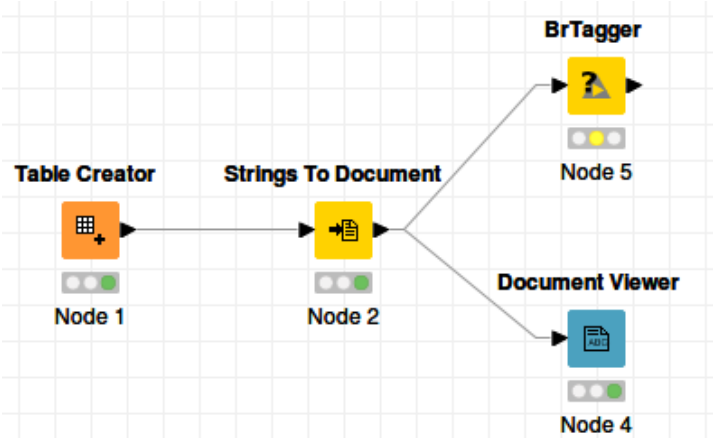



*Figura 1: Workflow inicial utilizado para testar o node BrTagger*

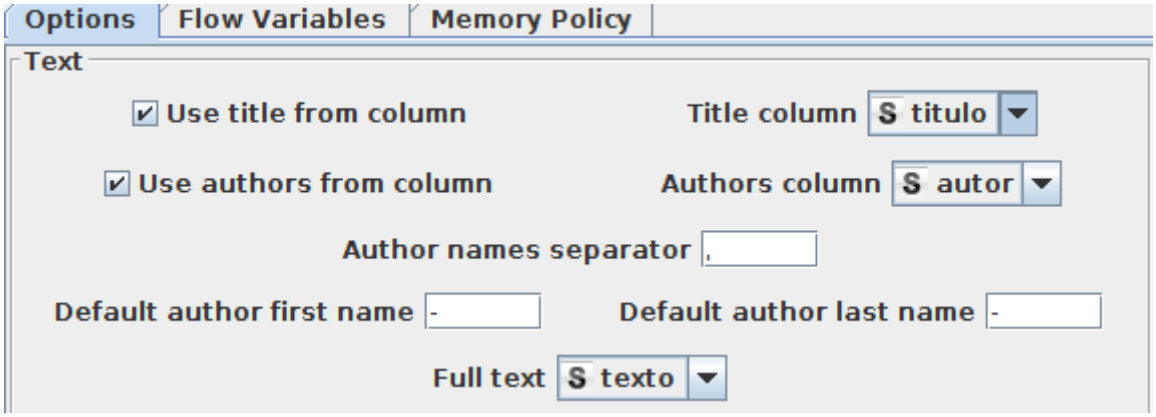
- No node **Table Creator**, criar uma tabela com a seguinte estrutura. Os dados são opcionais.

| Autor | Titulo | Texto |
|---|---|---|
| NomeAutor1 | Titulo1 | Isso é um exemplo de texto no documento 1. Este documento possui dois (2) parágrafos. |
| NomeAutor2 | Titulo2 | Este texto está presente no documento de número 2. Serão textos utilizados para análise morfossintática no idioma português. |
| NomeAutor3 | Titulo3 | Issu aki eh um testo escrito di forma totaumente errada pro POS Tagger naum conseguir clasificar as palavras de forma correta. Geralmente eh o tipo de testo que achamos muito em redes sociais, porque o brasileiro não sabe mais ler e muito menos escrever. |

- As configurações do node devem ficar similares à figura abaixo

| autor | titulo | texto |
|---|---|---|
| NomeAuto... | Titulo1 | Isso é um exemplo de texto no docum... |
| NomeAuto... | Titulo2 | Este texto está presente no documen... |
| NomeAuto... | Titulo3 | Issu aki eh um testo escrito di forma t... |

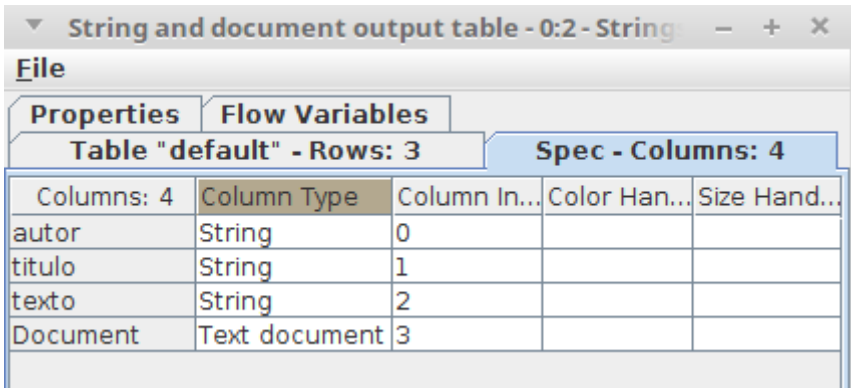
- Agora, configurar o node **Strings to Document**, indicando cada uma das colunas que devem servir de Título (*Title*), Autores (*Authors*) e Conteúdo (*Full text*):




- Executar os nodes **Table Creator** e **Strings to Document**. NÃO executar o BrTagger ainda!

- Salvar o Workflow e fechar o KNIME. Agora será desenvolvida a leitura do documento. Para isso, devemos limpar o código de exemplo gerado automaticamente pelo Eclipse.

- Após a execução do workflow, a tabela de saída do node Strings to Document deve mostrar algo similar à figura abaixo. **Lembrar que a coluna do documento é a coluna de índice 3**. Isto será importante na etapa inicial de codificação.



### 5.3.1 *"Limpando" o código*

- No arquivo BrTaggerNodeDialog.java, remover toda a "linha" de addDialogComponent, de forma que o código tenha apenas o método construtor:

```
public class BrTaggerNodeDialog extends DefaultNodeSettingsPane {

    /**
     * New pane for configuring BrTagger node dialog.
     * This is just a suggestion to demonstrate possible default dialog
     * components.
     */
    protected BrTaggerNodeDialog() {
        super();
    }
}
```

- Remover todas as variáveis (CFGKEY_COUNT, DEFAULT_COUNT e m_count). A parte superior do código fica desta forma:

```
public class BrTaggerNodeModel extends NodeModel {

    // the logger instance
    private static final NodeLogger logger = NodeLogger
            .getLogger(BrTaggerNodeModel.class);

    /**
     * Constructor for the node model.
     */
    protected BrTaggerNodeModel() {

        // TODO one incoming port and one outgoing port is assumed
        super(1, 1);
    }
```

- Remover todo o conteúdo do método **execute** e deixar o retorno como sendo o valor de inData, conforme figura abaixo:

```
@Override
protected BufferedDataTable[] execute(final BufferedDataTable[] inData,
        final ExecutionContext exec) throws Exception {

    return inData;
}
```

- Remover todas as referências de m_count nos métodos **saveSettingsTo**, **loadSettingsFrom** e **validateSettings**. Os métodos ficam “limpos”, conforme figura abaixo:

```
@Override
protected void saveSettingsTo(final NodeSettingsWO settings) {
}

/**
 * {@inheritDoc}
 */
@Override
protected void loadValidatedSettingsFrom(final NodeSettingsRO settings)
        throws InvalidSettingsException {
}

/**
 * {@inheritDoc}
 */
@Override
protected void validateSettings(final NodeSettingsRO settings)
        throws InvalidSettingsException {
}
```

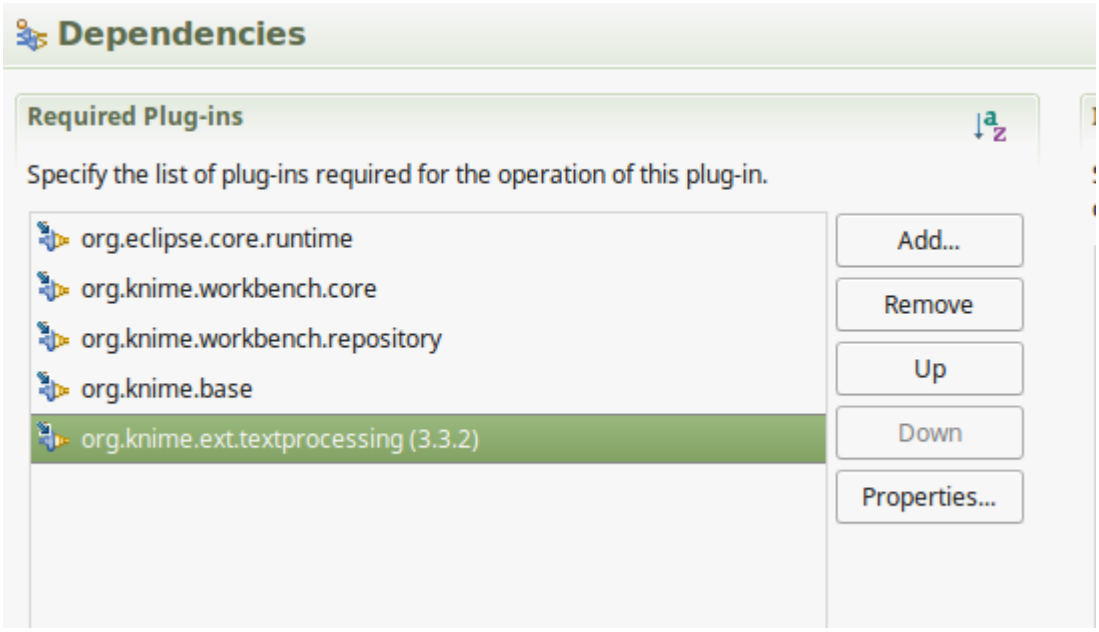
### *5.3.2 Lendo o documento (para os mais “apressados”)*

Com o código “limpo” é possível implementar a leitura do documento! A lógica dos nodes é executada pelo método **execute()**, da classe NodeModel (herdada neste exemplo pela classe BrTaggerNodeModel). Os métodos serão explicados à medida que for necessário ;)

- Definir o método *execute*() conforme código abaixo:

```
protected BufferedDataTable[] execute(final BufferedDataTable[] inData,
  final ExecutionContext exec) throws Exception {

  /* Indice da coluna do documento */
  final int colIndex = 3;

  RowIterator it = inData[0].iterator();
  while (it.hasNext()) {
    DataRow row = it.next();
    DataCell cell = row.getCell(colIndex);
    DocumentValue docv = (DocumentValue) cell;
    Document doc = docv.getDocument();
    String authors = doc.getAuthors().toString();
    String title = doc.getTitle();
    String txt = doc.getText();
    logger.info("Autores: " + authors);
    logger.info("Título: " + title);
    logger.info("Conteúdo: " + txt);
  }
  return inData;
}
```

- Incluir o contexto das classes do textprocessing (pacote org.knime.ext.textprocessing) no node. Para isso, é necessário editar o arquivo plugin.xml.
  - Pode ser pela interface gráfica do Eclipse conforme figura abaixo:




  - Ou incluir a linha destacada “na mão”, conforme trecho abaixo:

```
Manifest-Version: 1.0
Bundle-ManifestVersion: 2
Bundle-Name: BrTagger-Node extension for KNIME Workbench
Bundle-SymbolicName: org.knime.brtagger; singleton:=true
Bundle-Version: 1.0.0
Bundle-ClassPath: brtagger.jar
Bundle-Activator: org.knime.brtagger.BrTaggerNodePlugin
Bundle-Vendor: Daniel Silva
Require-Bundle: org.eclipse.core.runtime,
 org.knime.workbench.core,
 org.knime.workbench.repository,
 org.knime.base,
 org.knime.ext.textprocessing;bundle-version="3.3.2"
Bundle-ActivationPolicy: lazy
Export-Package: org.knime.brtagger
```

- Compilar o node e executar o KNIME
  - Run → Run Configurations → Selecionar o KNIME → Run
  - ou Ctrl + F11
  - **Não executar o node ainda, apenas o KNIME, que será aberto com o workflow da Figura 1**

- Acompanhar o log de execução

Antes de executar o node, abrir uma janela com terminal, shell, cmd, cygwin, console ou seja lá o que você usar para rodar comandos, para **acompanhar o arquivo de log** produzido pelo ambiente de execução. No linux, pode-se fazer isso com o comando **tail**.

Neste exemplo: o log de execução do knime "dentro" do Eclipse está no caminho definido na seção 4.1 (Criação do ambiente de execução) - Opção "Location".

- ${workspace_loc}/../runtime-KNIME/.metadata/knime/knime.log

- ${workspace_loc} = $HOME/Documentos/estudos/knime-dev/workspace
- um diretório acima = $HOME/Documentos/estudos/knime-dev
- Log = $HOME/Documentos/estudos/knime-dev/runtime-KNIME/.metadata/knime/knime.log

Para acompanhar o log:
tail -f $HOME/Documentos/estudos/knime-dev/runtime-KNIME/.metadata/knime/knime.log

- **Executar o node BrTagger (acompanhando o log!)**

As mensagens destacadas a seguir devem aparecer no log de execução do node:

2017-08-16 21:21:37,295 : DEBUG : KNIME-Worker-0 : LocalNodeExecutionJob : BrTagger : 0:5 : BrTagger 0:5 Start execute
... <várias e várias mensagens de log> ...
2017-08-16 21:21:37,298 : INFO : KNIME-Worker-0 : BrTaggerNodeModel : BrTagger : 0:5 : Autores: [org.knime.ext.textprocessing.data.Author@b1f712b8]
2017-08-16 21:21:37,298 : INFO : KNIME-Worker-0 : BrTaggerNodeModel : BrTagger : 0:5 : Título: Titulo1
2017-08-16 21:21:37,298 : INFO : KNIME-Worker-0 : BrTaggerNodeModel : BrTagger : 0:5 : Conteúdo: Titulo1Isso é um exemplo de texto no documento 1. Este documento possui dois (2) parágrafos.
2017-08-16 21:21:37,299 : INFO : KNIME-Worker-0 : BrTaggerNodeModel : BrTagger : 0:5 : Autores: [org.knime.ext.textprocessing.data.Author@b1f7135c]
2017-08-16 21:21:37,299 : INFO : KNIME-Worker-0 : BrTaggerNodeModel : BrTagger : 0:5 : Título: Titulo2
2017-08-16 21:21:37,299 : INFO : KNIME-Worker-0 : BrTaggerNodeModel : BrTagger : 0:5 : Conteúdo: Titulo2Este texto está presente no documento de número 2. Serão textos utilizados para análise morfossintática no idioma português.
2017-08-16 21:21:37,299 : INFO : KNIME-Worker-0 : BrTaggerNodeModel : BrTagger : 0:5 : Autores: [org.knime.ext.textprocessing.data.Author@b1f71400]
2017-08-16 21:21:37,299 : INFO : KNIME-Worker-0 : BrTaggerNodeModel : BrTagger : 0:5 : Título: Titulo3
2017-08-16 21:21:37,300 : INFO : KNIME-Worker-0 : BrTaggerNodeModel : BrTagger : 0:5 : Conteúdo: Titulo3Issu aki eh um testo escrito di forma totaumente errada pro POS Tagger naum conseguir clasificar as palavras de forma correta. Geralmente eh o tipo de testo que achamos muito

em redes sociais, porque o brasileiro não sabe mais ler e muito menos escrever.

CASO acontecer da execução do node falhar e da seguinte mensagem aparecer:

2017-08-16 21:17:32,730 : DEBUG : KNIME-Worker-0 : BrTagger : BrTagger : 0:5 : Execute failed: org/knime/ext/textprocessing/data/DocumentValue
java.lang.NoClassDefFoundError: org/knime/ext/textprocessing/data/DocumentValue
    at quintosdosinfernosaquelemontedeclassesmalditasdojava (arquivo.java:linha)
Caused by: java.lang.ClassNotFoundException: org.knime.ext.textprocessing.data.DocumentValue cannot be found by org.knime.brtagger_1.0.0

Significa que as classes do textprocessing não estão sendo encontradas e que provavelmente o passo de Incluir o contexto das classes do textprocessing não foi feito de forma correta.

ELABORAÇÃO EM ANDAMENTO. SEJA PACIENTE ;)