Bioinformatics and the UIMA SDK:

This is an exercise in programming and problem solving with technology relevant to Watson. The goal is to show your approach to problem solving, your ability to learn and leverage existing technology and your programming ability. We will use the code you write for this exercise to evaluate your ability to implement efficient algorithms and data structures. You should pay special attention to design, correctness and maintainability. Your presentation will be used to assess your ability to grasp complex concepts and solve problems.

The goal of this exercise is to develop a UIMA application that can transform DNA sequences into their protein sequence counterparts and compute the optimal alignment using a technique similar to computing Levenshtein or edit distance. Don't worry too much about the biology. This should be a simple exercise in dynamic programming and exploring the UIMA framework.

There are two main stages to this pipeline. The first stage generates the protein sequences and stores them in a UIMA CAS. The second stage of the pipeline reads each pair of protein sequences from the CAS, computes the optimal alignment and stores the results in the CAS.

The input is a collection of 5 DNA sequences taken from a popular bioinformatics benchmark. You need to write one UIMA annotator that generates all possible protein sequence pairs in order to do a pairwise alignment. There are instructions in the source code we provide for how to compute a protein sequence given a DNA sequence. You should refer to the DNA codon table on Wikipedia.

You need to write a second annotator that reads the output of the first annotator from the CAS and computes the optimal alignment of each pair of sequences and stores the result in the CAS. You should use a cost function such that the cost of an insertion or delete is 2 and the cost of a substitution is 1. An example of an alignment of the sequences '`AAGT`' and '`AGT`' is:

```
AAGT
A-GT
```

The '`-`' represents 1 insertion/delete. The total cost of this alignment is 2. Note that computing an alignment is equivalent to computing the minimum edit distance between two strings. There are simple algorithms for this (see Needleman-Wunsch) but we will be impressed by enhancements such as algorithms that use space linear in the length of the smallest sequence. We expect you to implement the algorithm directly. Please do not use an implementation from a 3rd party.

For extra credit, extend the alignment algorithm to work with more than 2 sequences. Extend the algorithm so that it may use a cost function that is more biologically plausible than our simple cost function and read this cost function from the UIMA CAS or resource. Compute the global alignment of all sequences. Compute a good lower bound of a global alignment and prove that it is a lower bound.

For extra extra credit, explain how this application can be extended to perform a clustering of the

sequences using ML techniques. Write the code for this and run on a larger number of sequences. Leverage the UIMA framework where appropriate.

This UIMA tutorial is a great place to start (NOTE: the downloads featured in this article contain an older version of the UIMA-SDK. Visit uima.apache.org for the latest release).
http://www.ibm.com/developerworks/webservices/tutorials/ws-uima/

We have provided a template UIMA application to start from. The source code includes helpful comments you should read before starting. You should use Eclipse with the UIMA plug-ins installed. The tutorial or the Apache UIMA website should provide sufficient instructions for installing and configuring Eclipse. You will need to update the classpath for the template project.

Present your results and include answers to the following questions:

What algorithms and data structures did you use in your application?

What can you say about the time and space complexity of your algorithms?

How could your application be improved or extended if you had more time?