Candidate Challenge: Python, NLP

The Problem

One of the common problems to tackle when dealing with natural language inputs is the infamous “typo” - spelling errors on the input we handle. Fortunately there are a wide array of existing solutions that help with spell checking. An engineer on our team handles integrating various available pieces together to form pragmatic solutions for various problems.

In this challenge, you are tasked with tokenising user input that may or may not contain spelling errors, preserving the raw (uncorrected) form of the source string and wrapping all this in a RESTful service. The following endpoints are required to be provided by service.

POST /tokenise
This should accept a JSON payload containing user input as shown below.

```
{
  "input": "The fox cn't jump over a fence"
}
```

Upon receiving the input, it is processed to generate a JSON response as shown below.

```
{
  "tokens": [
    {"token": "The", "pos": "DET", "raw": "The"},
    {"token": "fox", "pos": "NOUN", "raw": "fox"},
    {"token": "ca", "pos": "VERB", "raw": "cn't"},
    {"token": "n't", "pos": "ADV", "raw": "cn't"},
    {"token": "jump", "pos": "VERB", "raw": "jump"},
    {"token": "over", "pos": "ADP", "raw": "over"},
    {"token": "a", "pos": "DET", "raw": "a"},
    {"token": "fence", "pos": "NOUN", "raw": "fence"}
  ]
}
```

Requirements

- You should use the following components for their respective purposes:
  - SpaCy for tokenising input: https://spacy.io/
- Service should be RESTful with the following endpoints
- Should run under Python 3.6, but is not required to run on Python 2.7
- Correct input conditions should be checked for, and proper error responses must be returned
- The standard Python tool chain for installing dependencies and creating any entry points should be used
- You are free to use third party packages
- Unit and integration test suites should be included
- Test suites should generate coverage reports

Submission

You can submit your code in one of three ways. If submitting via email, you can send it as an attachment with your submission email, or you may send us a share link from Google Drive, Dropbox etc.

Git Bundle

You can use git-bundle command to create a bundle and submit it as an archive (tarball/zip).

See: https://git-scm.com/docs/git-bundle

Archive

You can make use of your CLI or system provided tools to create a tarball or a zip file of your local repository copy.

Remote Repository

If none of the above works for you, you may also provide us access to a remote copy of your repository on GitHub, GitLab, BitBucket etc. If using this method, please make sure the repository is either **private** or is **removed** once the interview process is completed. You can request a username to provide access to by contacting us.

Evaluation Criteria

Working Code

The code is expected to be working out of the box, and any dependencies and instructions required to run it documented clearly in a README file delivered with your submission.

Requirement Adherence

This is rather self-explanatory, if we expected to build a car - we do not expect a bus.

Code Quality

If you have worked on at least one already existing code project along the way, you will have noticed you read code more often than you write it. With this in mind, and some compassion to our team mates who evaluate your submission. Keep it clean, readable and commented as required.

Test Coverage

A good code base is expected to have at the very least decent unit test coverage. This means you need to write tests that cover functionality adequately and if possible even extend to end-to-end testing. Perhaps even a metric to indicate how good a job you have done.

Additional Criteria

Code History
One of the primary reasons we insist on a git-archive as you submission is that we would like to see how you to worked on the code base, and how it evolved. Any good developer, keeps their working branch up-to-date and the work committed. Here the purpose is not the content of the commit messages or how often you have "Fixed typo" in your commits, but rather how the code evolved.

Packaging
The python ecosystem has had a fair amount of changes when it comes to packaging, standards around it and tooling. We do not prescribe any particular

tooling, but we do expect your package to adhere to packaging standards when it comes to naming etc and also metadata and dependencies.