OUTLINE

1. *Goal* — What I wanted to accomplish;
2. *Motivation* — Why I wanted to accomplish the goal;
3. *Context* — Defines several jargon terms;
4. *Solution* — What I created;
5. *Validation* — How I validated if the target audience wants to use the proposal;
6. *Recommendations* — What went wrong and future work;
7. *Conclusion.*

RESEARCH OBJECTIVE

> *A generic documentation (the "specification") and example implementation (the "program") that exposes an interface (the "API") for the topic ("text manipulation") based on real use cases of potential users (the "developer") on the platform (the "web").*

RESEARCH QUESTION

> *How can a specification and program, that exposes an API for text manipulation, based on use cases of developers on the web, be developed?*

QUESTIONS?

SUMMARY

NLP covers many challenges. The process of accomplishing these challenges touches on well-defined stages, such as *tokenisation*, the focus of the proposal. Current implementations on the web platform are lacking. In part, because advanced machine learning techniques (such as *supervised learning*) do not work on the web.

The audience that benefits the most from better parsing on the web platform, are web developers, a group which is more interested in practical use, and less so in theoretical applications.

The target audience's use cases for NLP on the web are vast. Examples include automatic summarisation, sentiment recognition, spam detection, typographic enhancements, counting words, language recognition, and more.

The presented proposal is split into several smaller solutions. These solutions come together in a proposal: *Retext*, a complete natural language system. *Retext* takes care of parsing natural language and enables users to create and use plug-ins.

Parsing is delegated to *parse-latin* and others, which first tokenise text into a list of words, punctuation, and white space. Later, these tokens are parsed into a syntax tree, containing paragraphs, sentences, embedded content, and more. Their intellect extends several well known techniques.

The objects returned by *parse-latin* and others are defined by NLCST. NLCST defines the syntax for these objects. NLCST is designed in similarity to other popular syntax tree specifications.

The interface to analyse and manipulate these objects is implemented by textOM. textOM is created in similarity to other, for the target audience well-known, techniques.

The proposal was validated both by solving the audience's use cases with *Retext*, and by measuring the audience's enthusiasm for *Retext*. Use cases were validated by implementing many as plug-ins for *Retext*. The enthusiasm showed by the target audience on social networks, through e-mail, and social coding was positive.