Goal

Our goals:

- Use WebLicht to construct a **large** automatically annotated treebank.
- Use TüNDRA to visualize and search **large** treebanks.

Requirements

- Annotation services should scale up:
  - **Concurrency:** many simultaneous users
  - **Parallelism:** material should be processed in parallel when the resources are available
  - **Resource management:** services should have resource limits to avoid impacting other services
  - **Fairness:** large jobs should not perceptibly impact interactive jobs
- Visualization/search tools should scale up: `large' in automatic annotations is two or three orders of magnitude larger than `large' in manual or semi-automatic annotations.
- Provide user-friendly interfaces to construct annotation chains, monitor progress, and retrieve/view results.

WebLicht overview

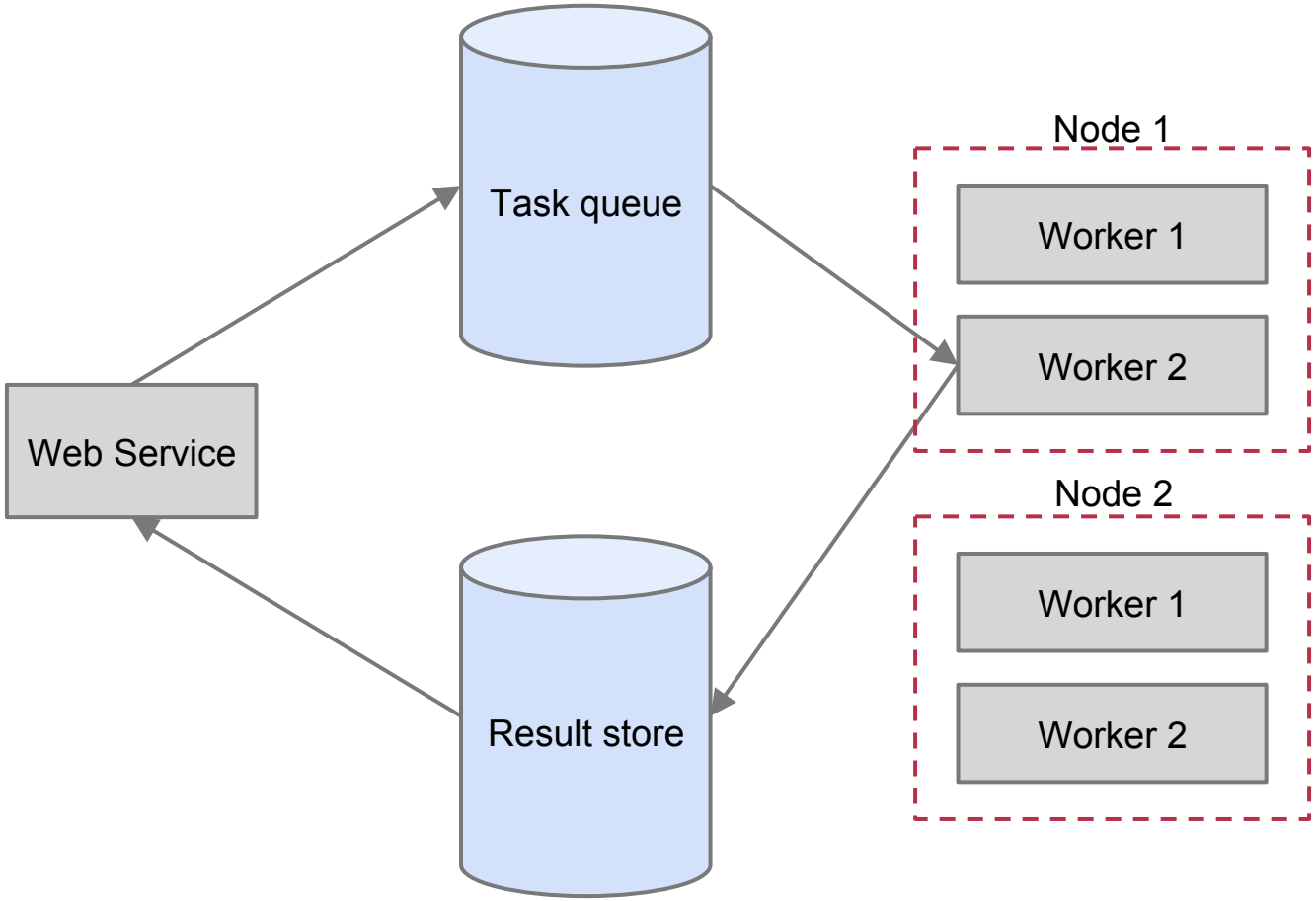
WebLicht is an execution environment for natural language processing pipelines:

- Uses a service-oriented architecture (SOA)
- Web services are combined to form a chain
- Chains are executed via sequential HTTP POST requests to services on the chain
- The output of service *n* is the input to service *n+1* in the chain
- Most services use Text Corpus Format (TCF) as their input and output
- Services add one or more annotation layer(s)

Consequences of the architecture

- Easy to add NLP tools to WebLicht:
  a. Make a RESTful wrapper service
  b. Create CMDI metadata
  c. Deploy service and metadata
- Scalability has to be implemented in the service.

Architecture overview



Distributed task queue

- Redis: distribute key-value store/data structure server
- Redis is used in various aspects of the architecture:
  - **Queues:** lists with atomic pop and put on in-flight list
  - **Result store:** lists with expiry, using the job UUID as the key
  - **Worker advertisement:** time-sorted sets
  - **Heartbeats:** key-value pairs with expiry

Distributed task queue (2)

- Most of the `glue' to use Redis as a distributed task queue is already provided by Jesque
- We modified Jesque to support durable queues (jobs can be recovered when a worker disappears)
- All other extra functionality is implemented as plugins (lifecycle listeners for Jesque):
  - Worker advertisement
  - Worker heartbeats
  - Result storage

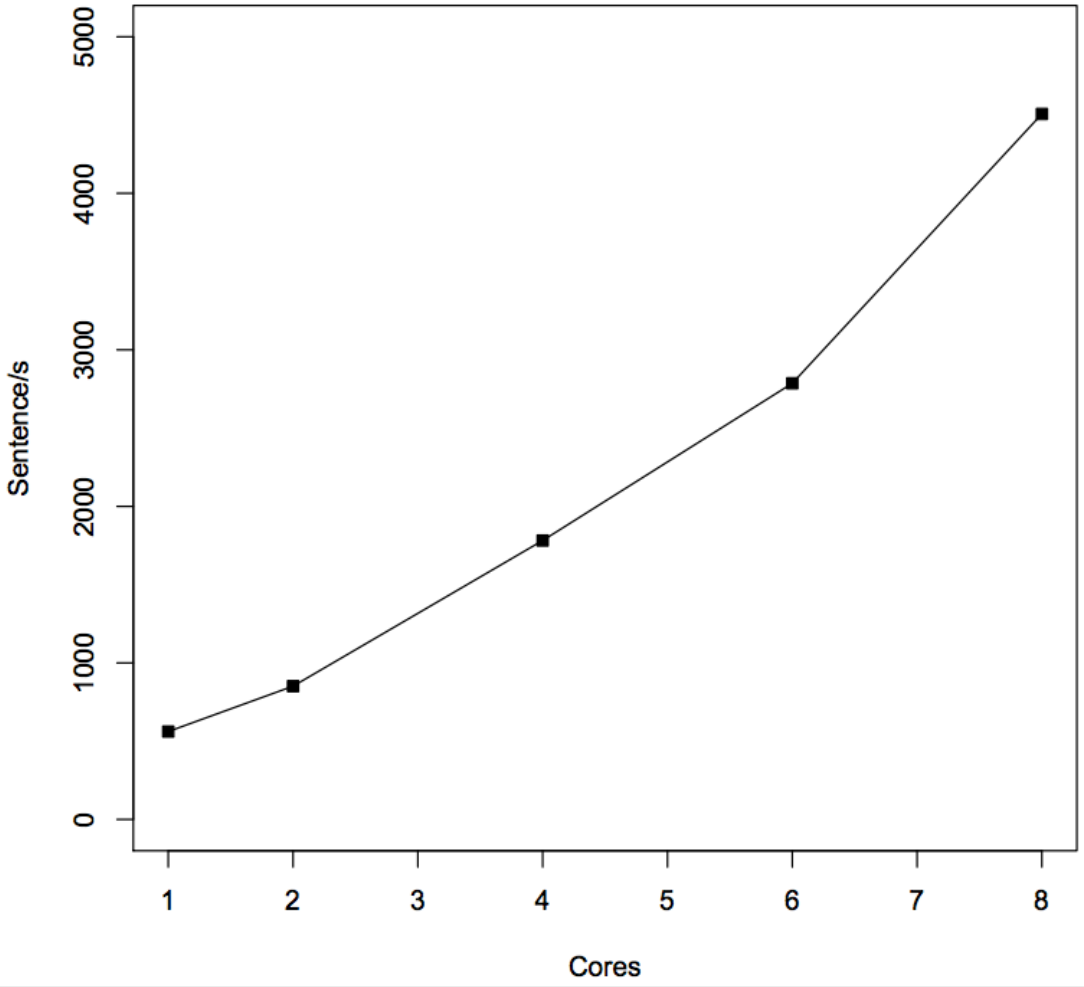
Introducing parallelism

- The distributed task queue provides concurrency: multiple clients can add jobs at the same time
- However, this does not provide parallel processing for a particular input
- We obtain parallelism by applying a TCF chunker:
  a. The chunker splits the TCF as chunks
  b. The chunks are submitted as separate jobs
  c. The chunker assembles result chunks
- Now workers can process chunks of an input in parallel

Fairness

- However, what happens if user A submits one year of newspaper text and user B submits one sentence?
- The chunks of user A's material block processing of that one sentence of user B
- To alleviate this problem, we introduce fairness:
  - Jesque workers can poll on multiple queues
  - Each queue is polled in turn
  - We create multiple queues for each service
  - We put the chunks for an input in the *n*th queue, where $n = log_{10}(|S|)$, where S is the size of the input.
- Result: chunks from small inputs get as much attention as chunks from large inputs

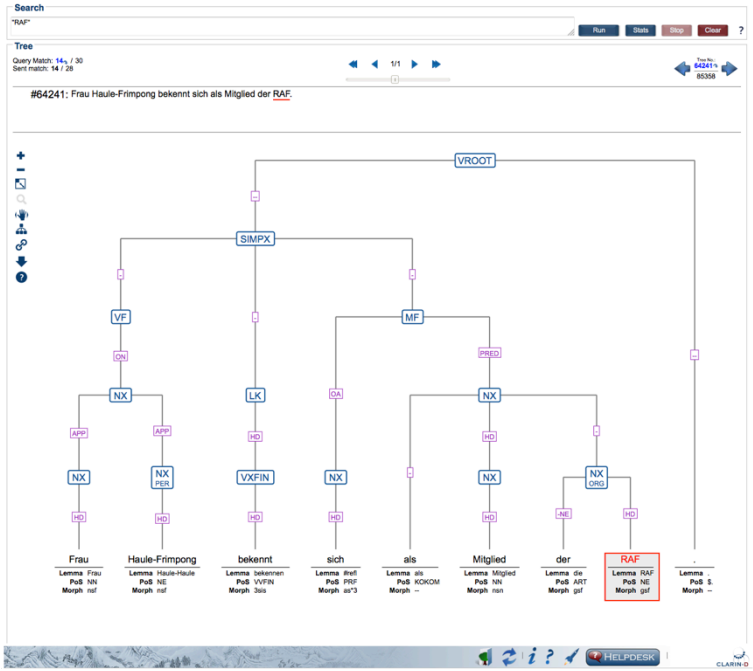
Evaluation



TüNDRA

CLARIN-D

TüNDRA is a web application for searching and visualizing treebanks.



- Originally developed for manually annotated treebanks
- Initial processing time for very large treebanks was too long when running a query
  - One large database index must be read from disk into memory

Solution:

- Split very large treebanks into many small ones which are queried successively
  - Reduces query preparation time to that of a small treebank
  - User sees results quickly

TüNDRA statistics

- Now shows intermediate results while a query is still running
- Allows a user see the general tendency
- Reservoir sampling is applied when queries produce a large number of results
  - Prevent running out of memory
  - Give reliable frequency estimates

Status

Done:

- Library for creating scalable WebLicht services
- Scalability of expensive parsing services:
  - Stanford parser (deployed)
  - Malt parser (deployed)
  - Berkeley parser (soon)
- TüNDRA now works with large (tens of million sentences) treebanks
- WebLicht REST API: WebLicht as a Service

Todo:

- An extension to WebLicht for long-running processes using WebLicht as a Service

Thank you!