POSIT: Simultaneously Tagging Natural and Programming Languages

Profir-Petru Pârțachi[†]

Santanu Kumar Dash[‡]

Christoph Treude[•]

Earl T. Barr[†]

† Department of Computer Science, University College London, London, United Kingdom
‡ Department of Computer Science, University of Surrey, Guildford, Surrey, United Kingdom
• School of Computer Science, University of Adelaide, Adelaide, South Australia, Australia

Mixed Text

On Fri, 24 Aug 2018 02:16:12 +0900 XXX <xxx@xxx.xxx>wrote:

[...]

Looking at the change that broke this we have:

<-diff removed for brevity>

Where “`real`” was added as a parameter to `__copy_instruction`.

Note that we pass in “`dest+ len`” but not “`real + len`” as your patch fixes. `__copy_instruction` was changed by the bad commit with:

<-diff removed for brevity->

[...]

Mixed Text

Where “`real`” was added as a parameter to `__copy_instruction`.

Mixed Text

ADV	**string_literal**	VERB	VERB	ADP	DET	NOUN	ADP	**method_name**	.
Where	“real”	was	added	as	a	parameter	to	__copy_instruction	.
English	Code	English	English	English	English	English	English	Code	English

POSIT: Segmenting and Tagging Mixed-Text

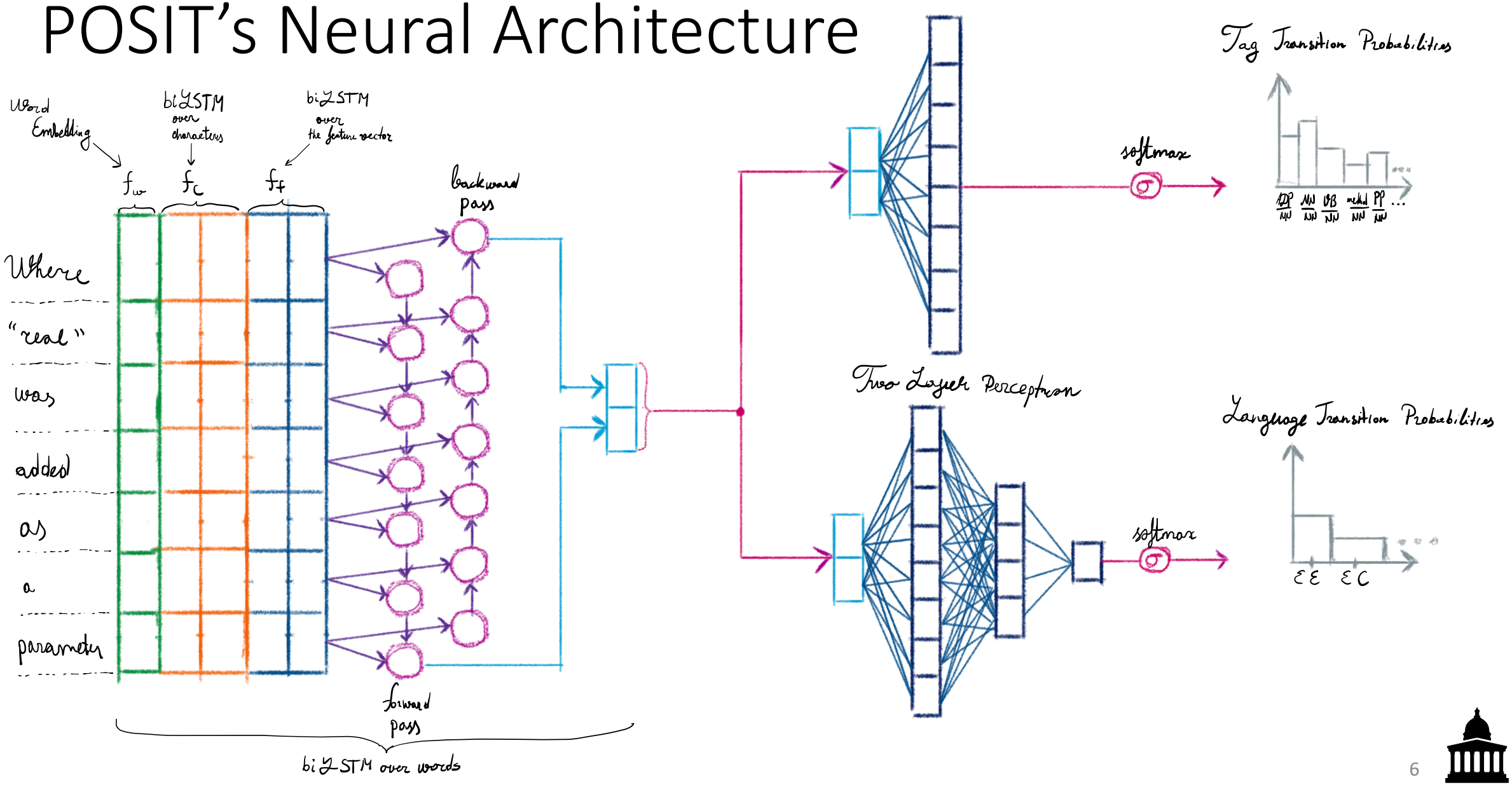
Mixed-text is ubiquitous in software development, but has mostly been handled as text in a single natural language.

POSIT solves the mixed-text tagging problem: given text with English and code, it segments it and tags it with AST or PoS tags.

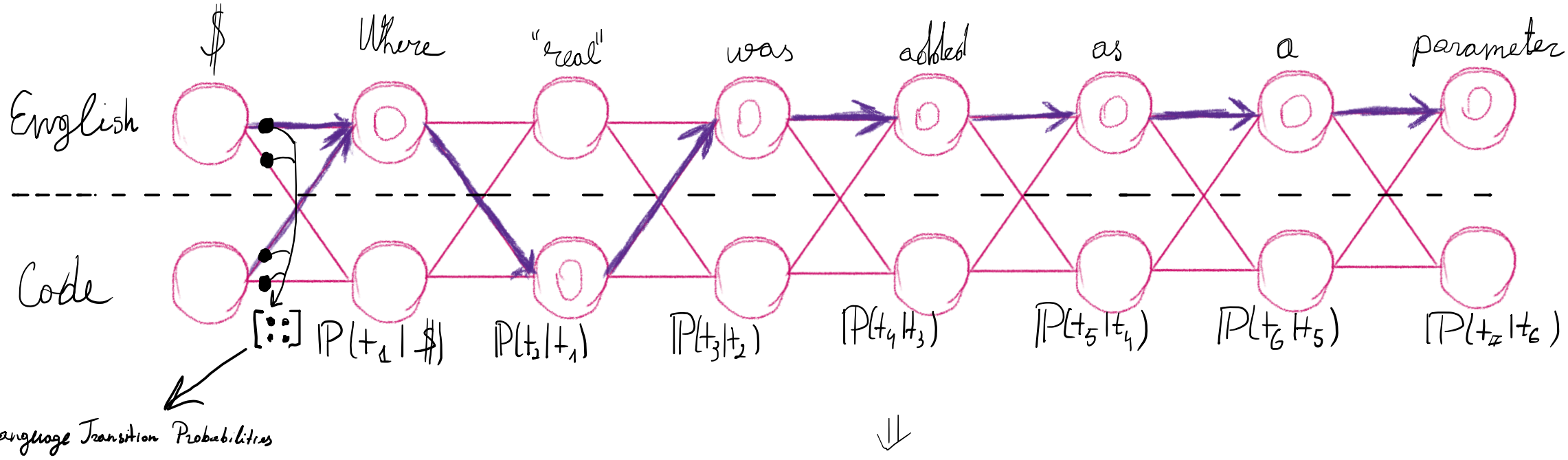
It is realised as a neural network trained on data from Stack Overflow and CLANG compilations.

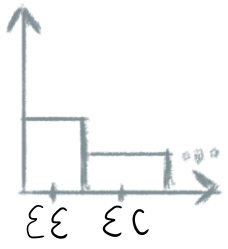
POSIT indirectly helps developers: it improves downstream tools on mixed-text: traceability, knowledge extraction, software artefact navigation, ontologies over mixed-text.

POSIT's Neural Architecture



Conditional Random Field (Viterbi Decode) Demonstrated on Language Segmentation





$P(t_t | t_{t-1})$

English	Code	English	English	English	English	English
Where	"real"	was	added	as	a	parameter

Experimental Set-up

We train POSIT on two corpora: Stack Overflow and Code Comments.

We compare POSIT with post-processed StORMeD output on Java Stack Overflow posts.

We augment TaskNav, a task extractor for software docs, with POSIT.

Corpus Statistics

Corpus Name	Tokens		Sentences		English Only Senteces		Code Only Sentences		Mixed Sentences	
	Train&Dev	Eval	Train&Dev	Eval	Train&Dev	Eval	Train&Dev	Eval	Train&Dev	Eval
Stack Overflow	7645103	2612261	214945	195021	55.8%	57.0%	32.6%	38.0%	11.6%	4.9%
Code Comments	132189	176418	21681	8677	11.3%	11.0%	79.4%	79.6%	9.4%	9.3%
Total	7777292	2788679	236626	203698	51.7%	55.1%	36.9%	39.7%	11.4%	5.1%

StORMeD as a Baseline

StORMeD is the pioneering work on mixed-text.

They use island grammars to separate Java code from English.

They provide a parsed Stack Overflow corpus, and a web service.

We adapt StORMeD to serve as a baseline for POSIT.

POSIT on Java Stack Overflow posts

Tool \ Task	Language Segmentation	Token Tagging
StORMeD	71.0%	61.9%
POSIT	81.6%	85.6%

We use Balanced Accuracy to assess the performance of the approaches.

POSIT on Stack Overflow and Code Comments

Corpus \ Task	Language Identification	Token Tagging
Stack Overflow	97.7%	93.8%
Code Comments	99.7%	98.9%
Mean	98.7%	96.4%

We use Balanced Accuracy to assess the performance of the different corpora.

TaskNav

Treude et al. built TaskNav to extract tasks from software documentation.

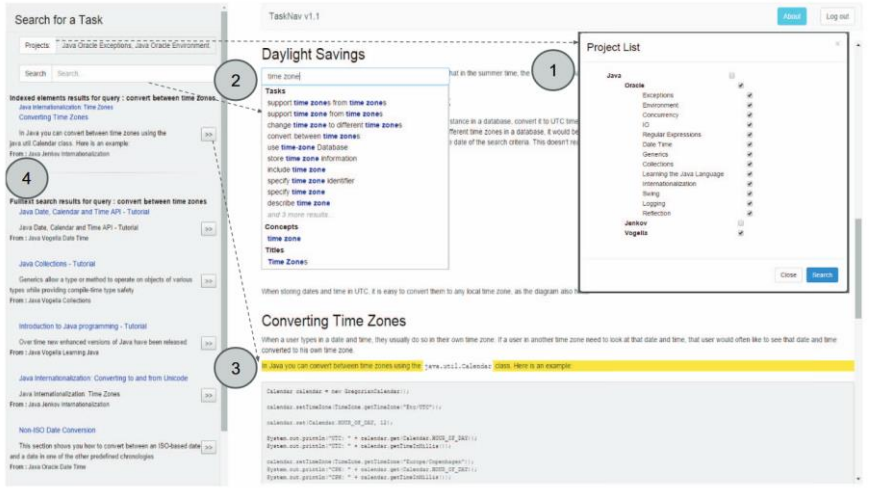


Fig. 1. TaskNav screenshot. (1) Project selection, (2) auto-complete, (3) highlighted search result, (4) full-text search results

It builds a dependency tree over part-of-speech tagged sentences that contain manually tagged code tokens.

Integrating POSIT into TaskNav

POSIT is a REST server: it takes a sentence and returns a tagging and segmentation.

TaskNav++ replaces TaskNav's segmenter and tagger with POSIT.

TaskNav++

Compared to TaskNav, TaskNav++ finds 97 new tasks over 30 e-mail threads.

Of these, manual annotators considered 65 (67.0%) considered reasonable.

TaskNav favours recall; TaskNav++’s 2.2 extra tasks improve recall.

Future Directions

POSIT aims to help developers indirectly: we hope it will help tools smiths and researchers produce better tools.

Traceability: POSIT's code-aware PoS tagging may improve precision.

Comprehension: POSIT separates code and English enabling code-sensitive and documentation-aware navigation.

Knowledge Extraction and Ontology: POSIT's segmenter facilitates separate analysis of the code and English in mixed-text.

Where to POSIT?

https://pppi.github.io/POSIT