
company-tabnine

melpa 20230216.817

TabNine is the all-language autocompleter. It uses machine learning to provide responsive, reliable, and relevant suggestions.

`company-tabnine` provides TabNine completion backend for company-mode. **It takes care of TabNine binaries**, so installation is easy.

```
int main(int argc, char *argv[]) {
    int n = 10;
    int* a = new int[n];
    bool* b = new bool[n];
    string* c = new string[n];
    new string[n];
    return 0;
}
new string[n] 24% 1
new string[n 25% 2
new string 25% 3
new string[n + 1]; 1% 4
new char 1% 5

def run_in_external_terminal(self):
    # TODO: Make this OS-specific.
    os_type = platform.system()
    if os_type == 'Linux':
        raise RuntimeError('Linux')
    Linux 50% 1
    with self.caster.lock: Linux is not supported 5% 2
    FILE_TEMPLATE = '#!' Linux is 17% 3 p "Press ENTER to
    TEMP_FILE = tempfil Linux is None: 17% 4
    with open(TEMP_FILE Linux does not support 2% 5
        f.write(FILE_TE Linux only 2% 6
        self.caster Linux systems 1% 7 .command))
    os.system('chmod +x Linux requires 1% 8
    os.system('open -a Linux is not 10% 9 _FILE))
    Linux currently 1% 0

class P4Command(Enum):
    HELP = P4Command(command='help', error_msg='Help Failed!')
    command=' 1
    command='help', error_msg='Help Failed!' 2
class P4Wrapper:
    command='help', error_msg='Help 3
    command='help 4
    class EnterRegte
    command='help', error_msg 5
    'help', 6
    def __enter__
    command='help', 7
    # TODO
    command='help', error_msg=' 8
    pass
    'help', error_msg='Help 9
    command 0

(defmacro spellcaster--define-renderer
  "Define a Spellcaster renderer w
  Define a function with BODY that a
  The first element of BODY can be t
  (declare (indent 2) (doc-string
  (if (not (fboundp ,name))
    (error "Function %s is not
  (cl-defun ,name (key is not
    ,@body))) is not
    not def
```

Installation

1. Make sure company-mode is installed and configured.
2. Install `company-tabnine`. This package is part of MELPA.

Note: See <https://melpa.org/#/getting-started> for MELPA usage. **Make sure to use the “bleeding-edge” repository instead of MELPA stable.**

-
- With `use-package`

Put the following in your config:

```
1 (use-package company-tabnine :ensure t)
```

- With `package.el` (built-in)

Install the package:

```
1 M-x package-install RET company-tabnine RET
```

Put the following in your config:

```
1 (require 'company-tabnine)
```

3. Add `company-tabnine` to `company-backends`

```
1 (add-to-list 'company-backends #'company-tabnine)
```

4. Run `M-x company-tabnine-install-binary` to install the TabNine binary for your system.

Recommended Configuration

Below are some recommended `company-mode` configuration that works well with `company-tabnine`.

```
1 ;; Trigger completion immediately.
2 (setq company-idle-delay 0)
3
4 ;; Number the candidates (use M-1, M-2 etc to select completions).
5 (setq company-show-numbers t)
```

Usage

`company-tabnine` should work out of the box.

See `M-x customize-group RET company-tabnine RET` for customizations.

Auto-balance parentheses

TabNine can automatically balance parentheses, by removing and adding closing parentheses after the cursor. See the examples here.

Note: The automatically-balancing happens in company's `post-completion` hook. However, `company-tng-frontend` actually suppresses this hook. In order to use automatic parentheses balancing, you need to manually call `company-complete-selection` or similar commands in this case, which will almost always happen if you do not use `company-tng-frontend`.

Known Issues

- `company-transformers` or plugins that use it (such as `company-flx-mode`) can interfere with TabNine's sorting. If this happens, put the following temporary workaround in your config:

```
1 ;; workaround for company-transformers
2 (setq company-tabnine--disable-next-transform nil)
3 (defun my-company--transform-candidates (func &rest args)
4   (if (not company-tabnine--disable-next-transform)
5       (apply func args)
6       (setq company-tabnine--disable-next-transform nil)
7       (car args)))
8
9 (defun my-company-tabnine (func &rest args)
10  (when (eq (car args) 'candidates)
11    (setq company-tabnine--disable-next-transform t))
12    (apply func args))
13
14 (advice-add #'company--transform-candidates :around #'my-company--
15             transform-candidates)
16 (advice-add #'company-tabnine :around #'my-company-tabnine)
```

- Spacemacs configurations can override the settings for `company-backends`.
- Conflict with ESS: See <https://github.com/emacs-ess/ESS/issues/955>
- TabNine's local deep learning completion might be enabled by default. It is very CPU-intensive if your device can't handle it. You can check by typing "TabNine::config" in any buffer (your browser should then automatically open to TabNine's config page) and disable Deep TabNine Local (you will lose local deep learning completion).