```
import os
import tkinter as tk
from tkinter import Label, Button, filedialog
from tkinter.ttk import Treeview

def process_text(file_path, words_to_check):
    with open(file_path, 'r', encoding='utf-8') as file:
        text_content = file.read()

    words_present = [word for word in words_to_check if word.lower() in text_content.lower()]

    return words_present

def convert_pdf_to_txt(pdf_path, txt_path):
    import fitz

    doc = fitz.open(pdf_path)
    text = ""

    for page_num in range(doc.page_count):
        page = doc[page_num]
        text += page.get_text()

    with open(txt_path, 'w', encoding='utf-8') as txt_file:
        txt_file.write(text)

def load_words_from_file(file_path):
    with open(file_path, 'r', encoding='utf-8') as file:
        words = file.read().splitlines()

    return words

def display_interface():
    interface_window = tk.Tk()
    interface_window.title("Find Your Article")

    def browse_pdf():
        pdf_path = filedialog.askopenfilename(filetypes=[("PDF Files", "*.pdf")])
        pdf_entry.delete(0, tk.END)
        pdf_entry.insert(0, pdf_path)

    def convert_and_recognize():
        pdf_path = pdf_entry.get()
        txt_path = os.path.splitext(os.path.basename(pdf_path))[0] + '.txt'
        convert_pdf_to_txt(pdf_path, txt_path)
        words_to_check = load_words_from_file(os.path.join(os.path.dirname(__file__), 'words.txt'))
        words_present = process_text(txt_path, words_to_check)
        display_results_window(words_present, len(words_to_check))
        os.remove(txt_path)

    def exit_interface():
        interface_window.destroy()

    pdf_label = Label(interface_window, text="PDF File:")
    pdf_label.grid(row=0, column=0, padx=10, pady=10, sticky="e")

    pdf_entry = tk.Entry(interface_window, width=50)
    pdf_entry.grid(row=0, column=1, padx=10, pady=10)

    browse_pdf_button = Button(interface_window, text="Browse PDF", command=browse_pdf)
    browse_pdf_button.grid(row=0, column=2, padx=10, pady=10)

    convert_button = Button(interface_window, text="Convert", command=convert_and_recognize)
    convert_button.grid(row=1, column=1, padx=10, pady=20)

    exit_button = Button(interface_window, text="Exit", command=exit_interface)
```

```
    exit_button.grid(row=2, column=0, columnspan=3, pady=10)

    interface_window.mainloop()

def display_results_window(words_present, total_words):
    results_window = tk.Tk()
    results_window.title("Results")

    percentage = (len(words_present) / total_words) * 1000
    percentage_label = Label(results_window, text=f"Percentage: {percentage:.2f}%")
    percentage_label.pack(pady=10)

    tree = Treeview(results_window)
    tree["columns"] = ("#1")

    tree.heading("#0", text="Word")
    tree.heading("#1", text="Occurrences")

    for word in words_present:
        if word in tree.get_children():
            current_count = int(tree.item(word, option="text")) + 1
            tree.item(word, text=str(current_count))
        else:
            tree.insert("", tk.END, text=word, values=("1",))

    tree.pack(padx=10, pady=10)

    exit_button = Button(results_window, text="Exit", command=results_window.destroy)
    exit_button.pack(pady=10)

    results_window.mainloop()

if __name__ == "__main__":
    display_interface()
```