FDIFF FAQ
Frank de Jong
04/10/2013

Why FDIFF?

I wrote FDIFF in 2009, when the office did not have any good tools for checking amended claim sets against original claim sets. Applicants sometimes send a working copy in amended form, but it turned out for me that these cannot be trusted. I decided to write a tool specifically aimed at comparing claim sets and presenting the changes in the best form for our work.

How does it work?

The standard algorithm "DIFF" for comparing two sets of text has been around for many years, and both the Google Diff and FDIFF are derivatives of it. The Diff algorithm basically compares two texts and finds the largest common part of text, giving a left-hand changed part, middle unchanged part and right-hand changed part for both texts. The algorithm then recursively finds, for the left-hand and right-hand part, which parts have changed and which have remained unchanged. Once it can't divide in changed and unchanged parts anymore, it codifies which parts have been removed, added, or have remained unchanged. See Example 1 below for an example.

FDIFF3 improves on the standard algorithm by specializing in finding the kinds of changes that are often used in amended claim sets. FDIFF4 improves on FDIFF3 by comparing separate claims against each other, rather than whole sets and identifying whether there is support for amendments in the original claims.

What is the difference between FDIFF and Google Diff?

In short, it's specialized for comparing claim sets. It does this by (a) splitting the claim set into individual claims and comparing claims with claims, rather than whole texts, (b) for those parts that were identified as having been added, it indicates the support in the original claim set through highlighting, (c) working on a word-level, (d) disregarding punctuation, whitespace and reference signs when comparing, (e) reformatting the claims in a readable form, (f) several small optimizations.

(a) It happens often that new claim 1 is based on an originally filed independent claim X (not being claim 1). The general DIFF algorithm then produces an output indicating either that the first X-1 claims have been deleted, or indicating that new claim 1 is completely new. By splitting up the old and new claim texts into individual claims, and comparing each new claims against each old claims, the output is much more in line with what you want to see: "New claim 1 is old claim 20 plus additions, minus deletions". FDIFF4 introduces this splitting.

(b) A very common occurrence is that new claim 1 is old claim X plus parts of claims Y and Z. FDIFF4 produces an output that gives you exactly this information. After having identified, for each new claim K, the original claim that is most like the new claim K, as well as the added parts and the deleted parts, the added parts are compared once again to the old claims and these added parts are highlighted with a particular color to indicate the old claim which has the most of this added part. See Example 3.

(c) When a claim changes wording from "said device" to "said devices", you want to really see that change, and you might miss it if it has only an underlined "s", as the standard Diff would codify it. FDIFF works at word level, and codifies that the word as a whole has changed. This also has a nice speed up effect, as fewer comparisons have to be made.

(d) When comparing amended claims with original claims, it happens often that punctuation marks, whitespace or reference signs are changed or introduced. If this happens, the standard DIFF algorithm fails to find the largest *practically* unchanged part, and would then start generating large blocks of deleted text and reappearing added text. See Example 2 below. FDIFF3 disregards changes like this when finding the largest common text, but does codify them in the end.

(e) When copy-pasting claim text from different sources (Viewer or OCR or Trimaran's "Dossier Full text" tab), line breaks are sometimes lost and the result is a huge blurp of text. FDIFF optionally reformats its output by inserting a line break behind the period ("."), and a line break and some indentation spaces after a semi-colon (";"). Sometimes this gives weird formatting, but most of the time it's useful.

(f) One small correction is that hyphens (generated by OCR) are translated into minus signs before texts are compared; this gives FDIFF a larger chance of finding the largest unchanged text. Another one is the disregarding of small words such as “the”, “a”, “of” when looking for support in original claims.

So will FDIFF always give exactly what I want?

In one word: no. Since the texts are often produced by OCR, 1s or Is are turned into ells, or 5s into Ss, commas into periods, etc. This will obviously have an impact on the comparison, and in particular the splitting of claims (which happens based on the claim number “1.”, “2.”, etc. If the output of FDIFF3 or FDIFF4 is not what you want, you can however simply edit the input texts manually and re-run the script.

Other cases might exist where applicants insert other characters that you might find uninteresting for the comparison. In that case I am open for suggestions.

Can I change the behaviour of FDIFF?

Yes, somewhat. You can turn on or off reformatting at the end, which is sometimes nice if you are not comparing claims. FDIFF4 usually works well without reformatting, FDIFF3 usually works best with reformatting.

Note by the way, that FDIFF4 defaults back to FDIFF3 behavior if no claims can be identified. So if you input simple test texts, the output you see will be FDIFF3.

Examples

Example 1.

Working example of basic DIFF algorithm:

"*This is a test line.*" compared to "*This is a silly line.*", the algorithm would find that "*This is a* " has remained unchanged, and generate empty left-hand parts, and right-hand parts "*test line.*" and "*silly line.*" It then recursively compares the two right-hand parts, finding that " *line.*" didn't change, and generating two left-hand parts "*test*" (deleted) and "*silly*" (added) and an empty right-hand part. Since there are no more common parts, the result is codified as 4 atoms: "This is a ", "~~test~~", "silly", " line.", which is output as: *This is a ~~test~~silly line.*

Example 2.

Example where normal DIFF goes wrong:

You can try this in the web page. Original text:

1. Apparatus for doing something with a first feature.
2. Apparatus according to claim 1, with a second feature.
3. Apparatus according to claim 1, with a third feature.

was changed to

1. Apparatus (1) for doing something with a first feature (2) and a third feature (4).
2. Apparatus according to claim 1, with a second feature (3).

In this case, the standard algorithm produces the following output:

1. Apparatus (1) for doing something with a first feature~~.~~
~~2. Apparatus according to claim 1, with a second feature.~~
~~3~~ (2) and a third feature (4).
2. Apparatus according to claim 1, with a ~~thir~~second feature (3).

Obviously, this is not what you want. Here it looks as though claim 2 has been removed, but in fact it didn't change at all, except for some reference signs. Also note how "third" changing to "second" is codified as though the "d" didn't change, which gives confusing information. In comparison, FDIFF3 produces the following output that shows right away that claim 3 was inserted into claim 1:

1. Apparatus (1) for doing something with a first feature (2) and a third feature (4).
2. Apparatus according to claim 1, with a second feature~~.~~
~~3. Apparatus according to claim 1, with a third feature~~ (3).

The difference is that FDiff recognizes that the text of claims 1 and 2 didn't change except for cosmetics, while the standard DIFF recognized "for doing something with a first feature" as the largest common text, and the first words of original claim 3 as the second largest, and then codified starting from there.

Example 3.

Example of FDIFF4 output

In comparison to the previous FDIFF3 example, FDIFF4 produces the following output (after inserting "great" before the "third feature" in both claim sets, to make sure that FDIFF4 finds the change large enough to even look for support):

New claim	Old claim	Difference
1	1 3	1. Apparatus (1) for doing something with a first feature (2) and a third great feature (4).
2	2	2. Apparatus according to claim 1, with a second feature (3).

Note how the yellow highlights indicate added items without support, while the cyan highlight indicates that support was found in old claim 3. Note also that FDIFF4 does not show the relatively uninteresting fact that claim 3 has been deleted. It just shows the new claims and how they relate to the old claims.