

# Esempi Overpass

## Contents

1. Farmacie a Palermo (uso di area, geocodearea e filtri di esistenza) .....	3
2. Parchi a Ravenna (uso di bbox, nwr e differenti tipi di out) .....	3
3. Via XXV Aprile a prescindere da come è scritta (uso di regexp) .....	3
4. Nodi con name:en o name:es (uso di regexp, velocità rispetto a filtro classico) .....	3
5. Nomi con parole che iniziano con minuscole (uso di regexp) .....	4
6. Tutti i luoghi di preghiera con il nome che include l'abbreviazione S. (uso di regexp) .....	4
7. CAP non corretti (uso di regexp).....	4
8. Nodi (effetto di > e <).....	4
9. Way (effetto di > e <) .....	5
10. Relazioni (effetto di >, <, >> e <<) .....	5
11. Tutti i nodi "fermata" di una linea bus (uso di node(r:platform))/ Il solo percorso di una linea bus (uso di way(r)) .....	6
12. Quali altre way intersecano i nodi di una way (uso di way(bn)).....	7
13. Quali linee bus usano una certa fermata (uso di rel(bn)) .....	7
14. Quali linee bus passano da una certa strada (uso di rel(bw)).....	7
15. La relazione route master che contiene la relazione di una linea bus (uso di rel(br)).....	8
16. Uso dei set.....	8
17. Block statements: difference .....	9
18. Intersezione .....	9
19. Block statements: con e senza foreach.....	10
20. Block statement: for.....	10
21. Block statement: IF .....	11
22. Block statement: complete.....	11
23. Block statement: retro .....	11
24. Block statement: compare .....	12
25. Filtri speciali: around semplice .....	12
26. Filtri speciali: around linestring.....	13
27. Filtri speciali: isclosed.....	13
28. Filtri speciali: isnumber e t[nometag] .....	13
29. Filtri speciali: length .....	14
30. Filtri speciali: version e user.....	14

31.	Filtri speciali: count_tags .....	14
32.	Filtri speciali: count_members.....	14
33.	Filtri speciali: count_by_role.....	14
34.	query is_in: comuni senza un amenity=townhall .....	14
35.	Dati dal passato: diff,adiff.....	16
36.	Dati in tabella.....	16
37.	Dati in tabella: differenza fra make e convert .....	16
38.	Dati in tabella: output con uso di foreach e map_to_area .....	17

## 1. FARMACIE A PALERMO (USO DI AREA, GEOCODEAREA E FILTRI DI ESISTENZA)

<https://overpass-turbo.eu/s/Qzf>

```
//{{geocodeArea:Palermo}}->.searchArea;

area[name=Palermo][admin_level=8][wikipedia="it:Palermo"]->.searchArea;

node[amenity=pharmacy](area.searchArea);

//node[amenity=pharmacy][!"opening_hours"](area.searchArea);

//node[amenity=pharmacy][!"opening_hours"][!name](area.searchArea);

out;
```

## 2. PARCHI A RAVENNA (USO DI BBOX, NWR E DIFFERENTI TIPI DI OUT)

<https://overpass-turbo.eu/s/Qzg> N.B. bbox non definito. Posizionarsi su Ravenna

```
nwr[leisure=park]({{bbox}});

//(. _;>);

out center;

//out skel;

//out;

//out meta;
```

## 3. VIA XXV APRILE A PRESCINDERE DA COME È SCRITTA (USO DI REGEXP)

<https://overpass-turbo.eu/s/Qzh> N.B. bbox non definito. Posizionarsi su Olgiate Comasco

```
way[name~"xv aprile",i]({{bbox}});

(. _;>);

out;
```

## 4. NODI CON NAME:EN O NAME:ES (USO DI REGEXP, VELOCITÀ RISPETTO A FILTRO CLASSICO)

<https://overpass-turbo.eu/s/Qzi> N.B. bbox non definito. Posizionarsi a nord di Milano

```
node[~"^name(:en|:es)$"~"."]({{bbox}});

//equivalente alla seguente, ma decisamente più veloce

//(

//  node["name:en"]({{bbox}});
```

```
// node["name:es"] ({{bbox}});  
// );
```

```
(. _;>);  
out;
```

## 5. NOMI CON PAROLE CHE INIZIANO CON MINUSCOLE (USO DI REGEXP)

<https://overpass-turbo.eu/s/QFm> N.B. bbox non definito. Posizionarsi su Lecco

```
[out:xml][timeout:25];  
way["highway"][name~"([a-z]|^[a-z])" ({{bbox}});  
(. _;>);  
out;
```

## 6. TUTTI I LUOGHI DI PREGHIERA CON IL NOME CHE INCLUDE L'ABBREVIAZIONE S. (USO DI REGEXP)

<https://overpass-turbo.eu/s/QFo> N.B. bbox non definito. Posizionarsi a sud di Brescia

```
[out:xml][timeout:25];  
nwr[amenity=place_of_worship][name~"^[^s]s\\. ",i] ({{bbox}});  
(. _;>);  
out;
```

## 7. CAP NON CORRETTI (USO DI REGEXP)

<https://overpass-turbo.eu/s/QFv> N.B. bbox non definito. Posizionarsi su Viterbo

```
[out:xml][timeout:25];  
nwr["addr:postcode"] ["addr:postcode"!~"^[0-9]{5}$"] ({{bbox}});  
(. _;>);  
out;
```

## 8. NODI (EFFETTO DI > E <)

<https://overpass-turbo.eu/s/Q4Y>

```
[out:xml][timeout:60];  
node(6186197500); //output=il nodo con i suoi riferimenti geografici  
//>; //output=niente. Non ho way o relazioni in input
```

```
//<; //output=le way che passano per quel nodo, ma senza riferimenti geografici
```

```
(. _;<;>;); //output=tutte le way passanti per il nodo ricercato e i loro nodi
```

```
out meta;
```

## 9. WAY (EFFETTO DI > E <)

<https://overpass-turbo.eu/s/Q4X>

```
[out:xml][timeout:60];
```

```
way(660737843); //output=way e nodi, ma senza riferimenti geografici
```

```
//>; //output=solo i nodi che la costituiscono
```

```
//<; //output=la way, i nodi della way, le altre way passanti per quei nodi e i loro nodi, ma senza riferimenti geografici
```

```
//(. _;>;); //output=tutti i nodi della way di partenza e le way passanti per essi (con i loro nodi)
```

```
//(. _;>;<;>;); //quanto sopra in forma contratta
```

```
out meta;
```

## 10. RELAZIONI (EFFETTO DI >, <, >> E <<)

<https://overpass-turbo.eu/s/Q4S>

```
[out:xml][timeout:60];
```

```
rel(9270000); //rel padre route master delle varianti della linea bus 3 a Pavia
```

```
//rel(9244636); //una rel membro del route master
```

```
//nodi, way della relazione stessa e delle relazioni membro
```

```
>>;
```

```
//relazione padre:niente (l'input set non contiene ne way, ne relazioni con membri di tipo way o node)
```

```
//relazione membro: nodi, way ma non i dati della relazione
```

```
//>;
```

```

//niente (l'input set non contiene ne nodi ne way)
//<;

//relazione padre:ritorna tutti i membri, ma non i dati geografici
//relazione membro:ritorna tutti i membri e la relazione che la contiene
(route master), ma non i dati geografici
//<<;

//relazione membro: ritorna tutte i dati anche della relazione master
che la contiene
//(<<;>>;);

out meta;

11. TUTTI I NODI "FERMATA" DI UNA LINEA BUS (USO DI
    NODE(R:PLATFORM))/ IL SOLO PERCORSO DI UNA LINEA BUS (USO DI
    WAY(R))
https://overpass-turbo.eu/s/QBr
[out:xml][timeout:60];

rel(3234153); //linea 25 Valle Salimbene-Pavia
//togliere il commento dalla seguente solo in caso si voglia avere in
output la relazione
>>;
//le sole fermate
//node(r:"platform");
//il solo percorso
//way(r);
//togliere il commento dalla seguente solo in caso si usi way(r)
//(. _i>;);

out;

```

## 12. QUALI ALTRE WAY INTERSECANO I NODI DI UNA WAY (USO DI WAY(BN))

<https://overpass-turbo.eu/s/QBt>

```
[out:xml] [timeout:60];
```

```
way(104911112);
```

```
> -> .a;
```

```
//.a out;
```

```
way(bn.a);
```

```
(._i>);
```

```
out;
```

## 13. QUALI LINEE BUS USANO UNA CERTA FERMATA (USO DI REL(BN))

<https://overpass-turbo.eu/s/QBu>

```
[out:xml] [timeout:60];
```

```
//fermata Stazione 4
```

```
node(1730353005);
```

```
rel(bn);
```

```
>>;
```

```
out;
```

## 14. QUALI LINEE BUS PASSANO DA UNA CERTA STRADA (USO DI REL(BW))

<https://overpass-turbo.eu/s/QBy>

```
[out:xml] [timeout:60];
```

```
//highway=residential Piazzale Stazione
```

```
way(44393383);
```

```
//relazioni linee autobus passanti per la via
```

```
rel(bw);
```

```
>>;
```

```
out;
```

## 15. LA RELAZIONE ROUTE MASTER CHE CONTIENE LA RELAZIONE DI UNA LINEA BUS (USO DI REL(BR))

<https://overpass-turbo.eu/s/QBw>

```
[out:xml][timeout:60];
```

```
//relazione linea bus 7B - MAUGERI MONDINO - PRADO
```

```
rel(3244581);
```

```
//relazione route master che la contiene
```

```
rel(br);
```

```
>>;
```

```
out;
```

## 16. USO DEI SET

<https://overpass-turbo.eu/s/QPG>

```
[out:json][timeout:25];
```

```
// memorizzo i parchi nel set .IMieiParchi
```

```
way["leisure"="park"]({{bbox}})->.IMieiParchi;
```

```
// memorizzo i parcheggi nel set .IMieiParcheggi
```

```
way["amenity"]({{bbox}})->.IMieiParcheggi;
```

```
// seleziono fra i parchi quelli con un nome. Il tutto finisce nel default set ._
```

```
way.IMieiParchi["name"];
```



```
.IMieiParcheggi out;  
.IMieiParcheggi >;  
out;
```

## 17. BLOCK STATEMENTS: DIFFERENCE

<https://overpass-turbo.eu/s/QbQ>

```
[out:xml] [timeout:60] [bbox:45.14,10.564,45.296,10.863];
```

```
way[highway=primary];
```

```
//(way[highway=primary]; - way(45.209,10.717,45.306,10.887));
```

```
(._;>);
```

```
out meta;
```

## 18. INTERSEZIONE

Nodi 1 <https://overpass-turbo.eu/s/Qzm> N.B. bbox non definito. Posizionarsi su Torino ad ovest di Porta Nuova

```
[out:xml] [timeout:60];  
node[tourism=hotel] ({{bbox}}) ->.hotels;  
node[internet_access] ({{bbox}}) ->.internet;  
node.hotels.internet;  
out;
```

Nodi 2 <https://overpass-turbo.eu/s/QBh> N.B. bbox non definito. Posizionarsi su Sarbogard a sud di Budapest

```
[out:xml] [timeout:60];  
way[highway] ({{bbox}});  
> -> .nodistrade;  
way[railway] ({{bbox}});  
> -> .nodiferrovie;  
node.nodistrade.nodiferrovie[!railway];  
out;
```

Way <https://overpass-turbo.eu/s/Qzn> N.B. bbox non definito. Posizionarsi a ovest di Salussola

//way di tipo residential che fanno parte di percorsi di hiking

```
[out:xml][timeout:60];
rel[route=hiking]({{bbox}});
(._;>)->.myhiking;
//
// con metodo intersezione
//
way[highway=residential]({{bbox}})->.myresidential;
way.myhiking.myresidential;
//
// con filtri
//
//way.myhiking[highway=residential];
//
(._;>);
out;
```

## 19. BLOCK STATEMENTS: CON E SENZA FOREACH

<https://overpass-turbo.eu/s/QBi> N.B. bbox non definito. Posizionarsi su qualche città con presenza di highway=secondary

```
way[highway=secondary]({{bbox}});
//foreach {
    (._;>);
    out;
//}
```

## 20. BLOCK STATEMENT: FOR

<https://overpass-turbo.eu/s/QJC> N.B. bbox non definito. Posizionarsi su Ferrara ad esempio

```
[out:csv(count,length,highway)];
way[highway]({{bbox}});
for (t["highway"])
```

```

{
  //_.val è il valore del tag t["highway"] all'interno del loop che
  cambia ad ogni iterazione
  make stat highway=_.val,
    count=count(ways),length=sum(length());
  out;
}

```

## 21. BLOCK STATEMENT: IF

<https://overpass-turbo.eu/s/Qi0>

```

//cerco un post_box nel raggio di 300 da un certo punto
node[amenity=post_box](around:300,51.178061,4.4214484);
//se non ne trovo allargo la ricerca a 1000 metri
if (count(nodes) == 0)
{
  node[amenity=post_box](around:1000,51.178061,4.4214484);
}

```

## 22. BLOCK STATEMENT: COMPLETE

<https://overpass-turbo.eu/s/QfZ>

```

//Tour dei pub a Dublino
[out:xml][timeout:60];
node(3470507586); // partiamo da qui...
//... e cerchiamo i pub ad una distanza di 500 metri dal nodo di partenza
e da ognuno dei pub trovati
//al massimo ne cerco 100
complete(100) { nwr[amenity=pub](around:500); };
out center;

```

## 23. BLOCK STATEMENT: RETRO

<https://overpass-turbo.eu/s/Qg6>

```

[out:csv(version,timestamp,changeset)];

```

```
//ottengo dei metaoggetti timeline che rappresentano le versioni
dell'oggetto (relazione con id 2632934) con diversi campi fra i quali
created (data creazione della versione)
```

```
timeline(relation,2632934);
```

```
//Per ogni valore del campo created del set generato da timeline...
```

```
for (t["created"])
```

```
{
```

```
  //...lo passo con _.val alla block statement retro
```

```
  retro(_.val)
```

```
{
```

```
  // l'output dell'oggetto relazione 2632934 è riferito alla data _.val
```

```
  rel(2632934);
```

```
  make
```

```
  stat
```

```
  version=u(version()),timestamp=u(timestamp()),changeset=u(changeset())
```

```
;
```

```
  out;
```

```
}
```

```
}
```

## 24. BLOCK STATEMENT: COMPARE

<https://overpass-turbo.eu/s/Qi6>

```
//dichiara le due date entro le quali si intende confrontare gli oggetti
```

```
[adiff:"2018-04-01T00:00:00Z","2018-05-01T00:00:00Z"];
```

```
area[name="Grenoble"]->.a;
```

```
node(area.a)[public_transport];
```

```
//compare con questa sintassi identifca i soli oggetti che fra le due
date indicate nei setting sopra hanno avuto il tag public_transport
cambiato
```

```
compare(delta:t["public_transport"]);
```

```
out meta;
```

## 25. FILTRI SPECIALI: AROUND SEMPLICE

<https://overpass-turbo.eu/s/Qj3>

```
[out:xml][timeout:25];
area[name="Belluno"]["admin_level"]=8];
//individuo tutte le fermate dell'autobus
node(area)[highway=bus_stop];
//trovo i parcheggi che stanno entro i 100 metri da queste
nwr(around:100)[amenity=parking];
(._;>);
out meta;
```

## 26. FILTRI SPECIALI: AROUND LINESTRING

<https://overpass-turbo.eu/s/Qj2>

```
[out:xml][timeout:25];
//tutti i parcheggi a 300m dalla circonvallazione interna di Milano
nwr["amenity"="parking"](around:300,45.48143815697844,9.18251037597656
,45.480354918187516,9.191608428955078,45.47409579738465,9.205169677734
377,45.46085305860481,9.207229614257812,45.452544661888965,9.201908111
572266,45.45326717976448,9.177017211914062,45.461695871036966,9.164314
270019531,45.47216977418839,9.166717529296873,45.47361429775643,9.1704
94079589844,45.47650323381734,9.167919158935547,45.47879020316614,9.16
929244995117,45.47879020316614,9.174613952636719,45.4762624948022,9.17
684555053711,45.4779476463103,9.180965423583984,45.48143815697844,9.18
1995391845703);

out meta;
```

## 27. FILTRI SPECIALI: ISCLOSED

<https://overpass-turbo.eu/s/QtU> N.B. bbox non definito. Posizionarsi su Firenze centro

```
way[highway=pedestrian](if:is_closed())(bbox);
(._;>);
out;
```

## 28. FILTRI SPECIALI: ISNUMBER E T[NOMETAG]

<https://overpass-turbo.eu/s/QtQ> N.B. bbox non definito. Posizionarsi su Laterina a nord-ovest di Arezzo

```
way[maxspeed](if:!is_number(t["maxspeed"]))(bbox);
(._;>);
out;
```

## 29. FILTRI SPECIALI: LENGTH

<https://overpass-turbo.eu/s/QtR>

```
rel["route"="hiking"](if:length()<500)(43.462061888033,11.513671875,43.473165573811,11.536438465118);

(._;>);

out;
```

## 30. FILTRI SPECIALI: VERSION E USER

<https://overpass-turbo.eu/s/QtS>

```
node(if:version("")==1&&user("")=="IlBano")(45.13584276099,9.1179227828979,45.141238792494,9.1293060779572);

out meta;
```

## 31. FILTRI SPECIALI: COUNT\_TAGS

<https://overpass-turbo.eu/s/QtT>

```
way(40.197953701303,16.593818664551,40.244681238557,16.684885025024)(if:count_tags()==0);

out geom;
```

## 32. FILTRI SPECIALI: COUNT\_MEMBERS

<https://overpass-turbo.eu/s/QtV>

```
way(if:count_members()>1900)(43.600284023536,18.717269897461,43.953776360127,19.44580078125);

(._;>);

out;
```

## 33. FILTRI SPECIALI: COUNT\_BY\_ROLE

<https://overpass-turbo.eu/s/QtW>

```
rel["type"="multipolygon"](if:count_by_role("outer")<1)(43.138071875541,4.691162109375,43.848393764892,6.1482238769531);

(._;>);

out;
```

## 34. QUERY IS\_IN: COMUNI SENZA UN AMENITY=TOWNHALL

<https://overpass-turbo.eu/s/Qkg>

```
//
```

```

//Quali comuni non hanno un amenity=townhall ?
//
[out:xml][timeout:90];
{{geocodeArea:Piemonte}}->.searchArea;

// memorizzo in allAdminBoundary tutti gli administrative boundary di
tipo comune come oggetti di tipo relation
relation["boundary"="administrative"]["admin_level"]=8] (area.searchArea
)->.allAdminBoundary;

//estraggo tutti i possibili elementi con tag amenity=townhall e li
memorizzo nel set features
(
  node["amenity"="townhall"] (area.searchArea);
  way["amenity"="townhall"] (area.searchArea);
  relation["amenity"="townhall"] (area.searchArea);
)->.features;

//recurse down per "espandere" tutte le relation e le ways con
memorizzazione in set a
(.features; >)->.a;

//memorizzo nel set b tutte le aree che contengono i dati memorizzati
nel set a. Ottengo dei meta-oggetti area perchè l'operatore is_in ritorna
solo questo tipo
.a is_in ->.b;

//filtro gli oggetti area cercando solo quelli di tipo comune e li
memorizzo nel set c
area.b[admin_level=8]->.c;

//con rel(pivot.c) riottengo le relazioni e le memorizzo nel set
withFeature. Questo passaggio è necessario per ottenere oggetti con
tipologia omogenea rispetto a quelli contenuti in allAdminBoundary

```

```
rel (pivot.c)->.withFeature;
```

```
//computo la differenza fra i due set ottenendo quindi tutte quelle aree  
di tipo admin_level=8 senza un townhall
```

```
(.allAdminBoundary; - .withFeature);
```

```
node (r:"admin_centre");
```

```
//(._;>);
```

```
out meta;
```

### 35. DATI DAL PASSATO: DIFF,ADIFF

<https://overpass-turbo.eu/s/QqY>

```
[out:xml] [timeout:90]
```

```
[adiff:"2018-09-14T15:00:00Z","2019-09-14T15:00:00Z"];
```

```
way(42.021019789303,13.411345481873,42.023861189367,13.417037129402);
```

```
out;
```

### 36. DATI IN TABELLA

<https://overpass-turbo.eu/s/QiZ>

```
[out:csv (::id, ::timestamp, ::user, ::type, "name", "fee"; true; ", ")] [timeou  
t:25];
```

```
nwr["amenity"="parking"] (45.387239028766,11.855964660645,45.4087247245  
74,11.901497840881);
```

```
out meta;
```

### 37. DATI IN TABELLA: DIFFERENZA FRA MAKE E CONVERT

<https://overpass-turbo.eu/s/Qyp>

```
[out:csv (::id,name,piedi,metri; true; ", ")];
```

```
{{geocodeArea:Cagliari}}->.searchArea;
```

```
way["highway"="secondary"] (area.searchArea);
```

```
//convert stat  
::id=id(),piedi=(length()/1000*0.62*5280),metri=length(),name=t["name"  
];
```



```
//make stat piedi=sum(length()/1000*0.62*5280),metri=sum(length());  
out;
```

### 38. DATI IN TABELLA: OUTPUT CON USO DI FOREACH E MAP\_TO\_AREA

<https://overpass-turbo.eu/s/QJH>

```
[out:csv(::type, "name", ::count, ::"count:nodes", ::"count:ways",  
::"count:relations")];
```

```
area["name"="Sondrio"]["admin_level"=6]->.ProvinciaSO;
```

```
rel(area.ProvinciaSO)["admin_level"=8];
```

```
map_to_area;
```

```
foreach->.Comune(
```

```
  // output del nome del comune
```

```
  .Comune out;
```

```
  // elementi leisure=park presenti nell'area .Comune
```

```
  nwr(area.Comune)[leisure=park];
```

```
  // Conteggio degli elementi in output
```

```
  out count;
```

```
);
```