Relatório de treinamento language model:

* 1º. Treinamento
  + No. de epochs: 1 + 15
  + Arquivo do modelo: lm\_PT\_v2\_3
  + Hiperparâmetros:
    - Vocabulário de 60000 tokens
    - lr=1e-3
    - em\_sz,nh,nl = 400,1150,3
    - bs = 20
    - learner.clip = 0.2
    - drops = np.array([0.25, 0.1, 0.2, 0.02, 0.15])\*0.7
    - wd=1e-7
    - bptt=70
    - Adam, betas=(0.8, 0.99)
  + Resultados:

epoch trn\_loss val\_loss

0 4.441398 4.394683

1 4.446199 4.436527

2 4.487567 4.380986

3 4.498426 4.342901

4 4.53838 4.322014

5 4.472949 4.295012

6 4.457945 4.287099

7 4.452684 4.27187

8 4.451175 4.255512

9 4.436211 4.242401

10 4.401264 4.242377

11 4.419634 4.218644

12 4.396261 4.209278

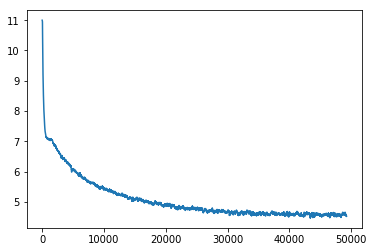
13 4.339406 4.199327

14 4.475429 4.181232

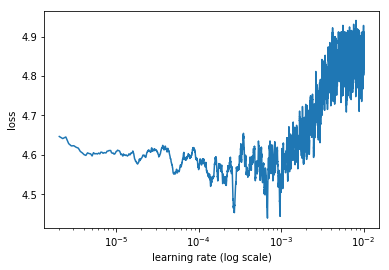
* + OBS.: Não salvou as acurácias
* 2º. Treinamento
  + No. de epochs: 1
  + Arquivo do modelo: lm\_PT\_v2\_fastai
  + Hiperparâmetros:
    - Vocabulário de 60000 tokens
    - lr=1e-3/2
    - em\_sz,nh,nl = 400,1150,3
    - bs = 32
    - drops = np.array([0.25, 0.1, 0.2, 0.02, 0.15])\*0.7
    - wd=1e-7
    - bptt=70
    - Adam, betas=(0.8, 0.99)
  + Resultados:

epoch trn\_loss val\_loss accuracy

0 4.523524 4.408578 0.261878



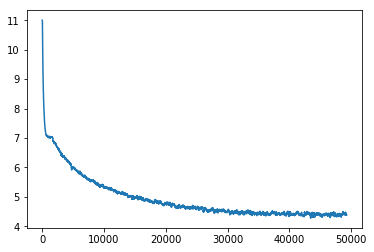
* + Resultado do learning rate finder: (melhor taxa aprox. 1e-3/2, conforme foi utilizado no treinamento da primeira epoch)



* 3o. treinamento:
  + No. de epochs: 1
  + Arquivo do modelo: lm\_PT\_v2\_fastai\_dp\_005
  + Hiperparâmetros:
    - Vocabulário de 60000 tokens
    - lr=1e-3
    - em\_sz,nh,nl = 400,1150,3
    - bs = 32
    - drops = np.array([0.25, 0.1, 0.2, 0.02, 0.15])\*0.05
    - wd=1e-7
    - bptt=70
    - Adam, betas=(0.8, 0.99)
  + Resultados:

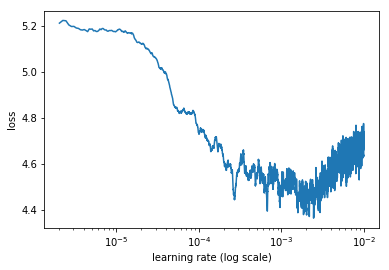
epoch trn\_loss val\_loss accuracy

0 4.394011 4.73682 0.260517



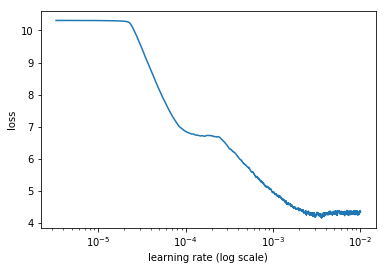
*Comentário: Embora não seja possível afirmar, mas o modelo anterior pode ter apresentado underfitting, enquanto este último modelo pode estar com overfitting. Pode ser necessário, portanto, aumentar a learning rate (regularização).*

* + Resultado do learning rate finder: (melhor taxa de aprox. 1e-3, ou seja, o dobro da anterior, confirmando a hipótese apresentada no comentário acima)

**

A próxima rodada de testes repetiu os parâmetros do 3º. Treinamento (e posteriormente os parâmetros posteriores) com o foco em acelerar o treinamento (vocabulário menor, bbtt menor, etc.)

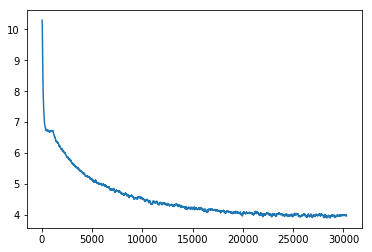
* 4o. treinamento:
  + Pré-treinamento “simulado” para encontrar a melhor taxa de aprendizagem
  + No. de epochs: 1
  + Hiperparâmetros:
    - Vocabulário de 30000 tokens
    - start\_lr=lrs/10, end\_lr=lrs\*10, linear=True
    - em\_sz,nh,nl = 400,1150,3
    - bs = 52
    - wd=1e-7
    - bptt=70
    - drops = np.array([0.25, 0.1, 0.2, 0.02, 0.15])\*0.05
    - Adam, betas=(0.8, 0.99)
  + Resultado do learning rate finder: (melhor taxa de aprox. 1e-3)



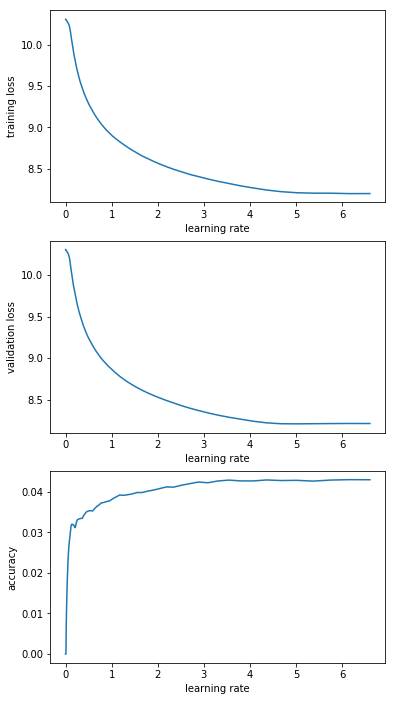
* + Resultados do treinamento (1 epoch):

epoch trn\_loss val\_loss accuracy

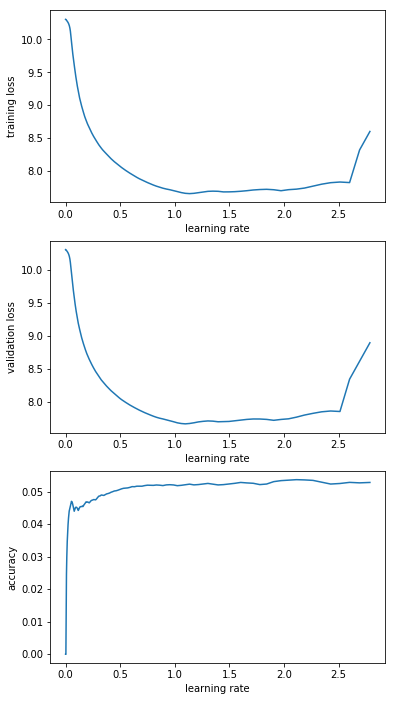
0 3.967844 3.933865 0.28279



* + Arquivo do modelo: lm\_PT\_v2\_fastai\_dp\_005\_30k\_tokens\_bs52
* 5o. treinamento:
  + Pré-treinamento “simulado” para encontrar a melhor taxa de aprendizagem
  + No. de epochs: 2
  + Hiperparâmetros:
    - Vocabulário de 30000 tokens
    - learner.lr\_find2(num\_it=200) (200 iterações até encontrar a melhor taxa)
    - em\_sz,nh,nl = 400,1150,3
    - bs = 52
    - wd=1e-7
    - bptt=70
    - drops = np.array([0.25, 0.1, 0.2, 0.02, 0.15])\*0.05
    - SGD, momentum=0.9
  + Resultado do learning rate finder: (melhor taxa de aprox. 5)



OBS.: A execução acima considerou o número máximo de iterações igual a 200. Aparentemente, se aumentar o número de iterações, dá a entender que a melhor taxa será maior que 5. Porém, ao fazer isso, considerando 400 iterações, os gráficos abaixo apontam o valor 1 para a taxa, com um erro de valor semelhante. Assim, optei pelo maior valor (5) para o treinamento subsequente.

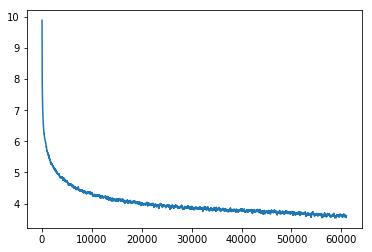


* + Treinamento por 2 epochs
    - use\_clr\_beta=(10,10,0.95,0.85)
  + Resultados do treinamento:

epoch trn\_loss val\_loss accuracy

0 3.812938 3.917015 0.273221

1 3.622323 3.618101 0.306443



* + Arquivo do modelo: lm\_PT\_1\_cycle
* 6º. Treinamento:
  + No. de epochs: 10
  + Hiperparâmetros:
    - em\_sz,nh,nl = 400,1150,3
    - bs = 52
    - wd=1e-7
    - bptt=70
    - drops = np.array([0.25, 0.1, 0.2, 0.02, 0.15])\*0.05
    - SGD, momentum=0.9
    - lr = 5.0
  + Resultado do treinamento:

epoch trn\_loss val\_loss accuracy

0 3.869934 3.955343 0.270151

1 3.683968 3.772445 0.28535

2 3.64038 3.706181 0.291559

3 3.565753 3.672791 0.293965

4 3.550975 3.648409 0.296805

5 3.438498 3.625634 0.299774

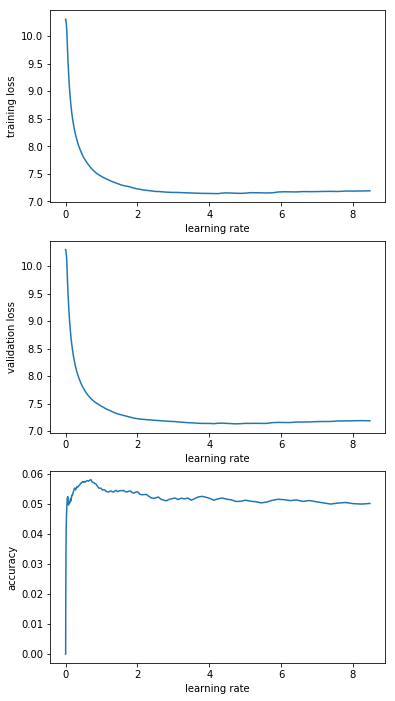
6 3.434014 3.599491 0.302151

7 3.42089 3.569387 0.305828

8 3.328981 3.51831 0.313064

9 3.341955 3.465898 0.320812

* + Arquivo do modelo: lm\_PT\_1\_cycle\_10\_epochs\_from\_scratch
* 7º. Treinamento:
  + Pré-treinamento “simulado” para encontrar a melhor taxa de aprendizagem
  + No. de epochs: 2
  + Hiperparâmetros:
    - em\_sz,nh,nl = 400,1150,3
    - bs = 52
    - wd=1e-7
    - bptt=70
    - drops = np.array([0.25, 0.1, 0.2, 0.02, 0.15])\*0.1
    - SGD, momentum=0.9

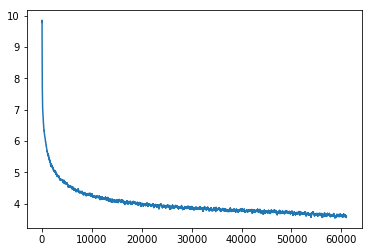


* + Treinamento por 2 epochs
    - use\_clr\_beta=(10,10,0.95,0.85)
  + Resultados do treinamento:

epoch trn\_loss val\_loss accuracy

0 3.808152 3.901497 0.273225

1 3.604043 3.611512 0.306141



* + Arquivo do modelo: lm\_PT\_1\_cycle\_dp\_gc